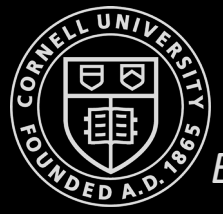
ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots
Graph Search



1 Answer

Kirstin Petersen STAFF
3 hours ago

I'd be very sorry to see you drop the class after putting in all this effort to get so far!

In fact, our observations this weeks indicates that many students are still working on the PID lab, instead of the Kalman filter lab. Of course, you should do whatever you think is best for you, but first consider one of these options instead:

1. Skip the KF lab entirely. You loose 7.5 points, which is hardly the end of the world. You can still complete lab 8 either by 1) extrapolating from the TOF measurements you actually get, or 2) by trying to manually tune the distance at which you start your turn/do your flip.

2. Skip the KF lab for now, and use one of your 2 unlimited time lab extensions to do KF later in the semester, so that you still get back the 7.5 points. (Be sure to let us know if that is the case).

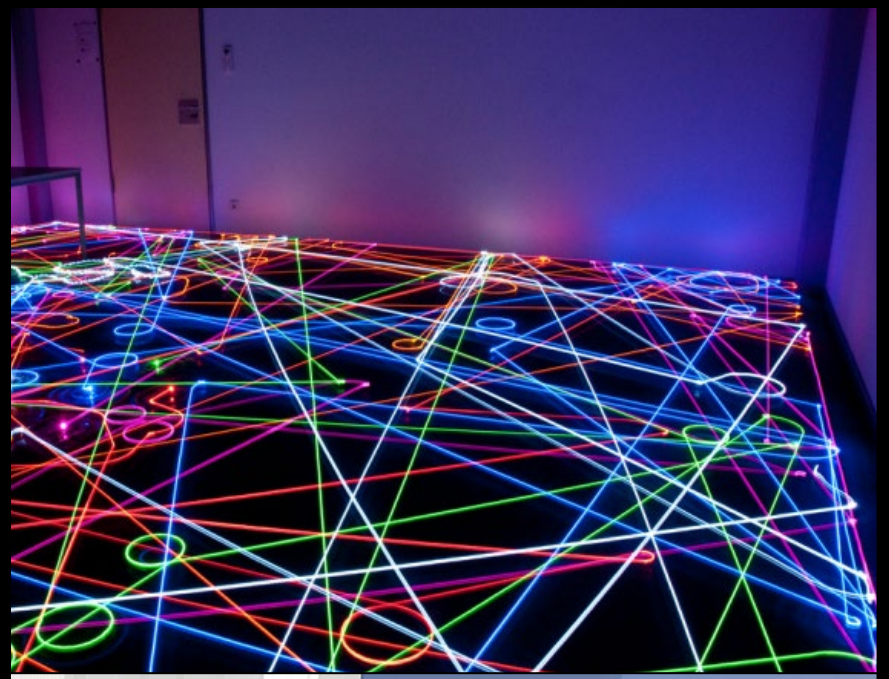
3. Do half the KF lab for half the points. This only involves a step response (to be done in the lab), and KF on the previous data set you got from your PID control.

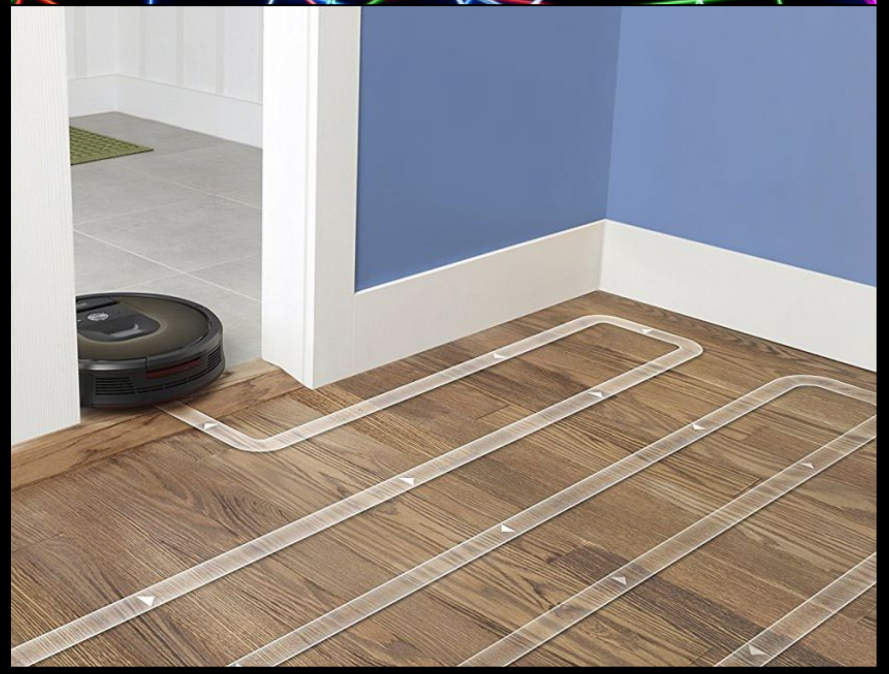
With respect to the PID lab itself, I'm happy to give an extension on the write-up if you think that would help! We can also discuss switching to audit later if you want to give it a few more weeks of thought. I can always sign a petition to allow you to switch after the drop deadline.

Comment Edit Delete Endorse •••

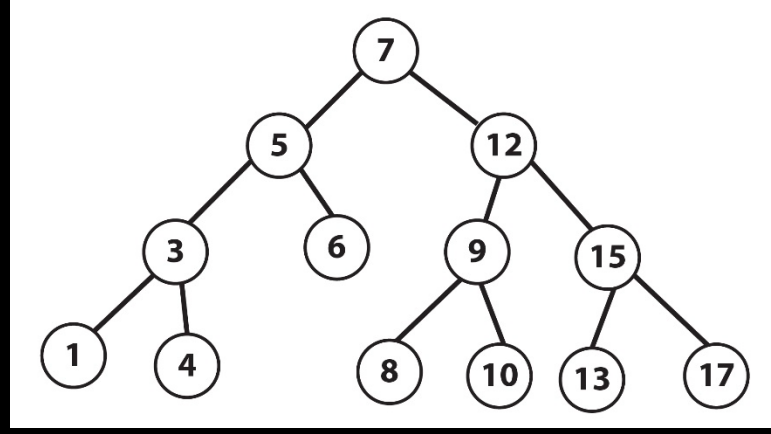
Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Uninformed search
    - Depth First Search (DFS)
    - Breadth First Search (BFS)
    - Dijsktra's Search
  - Informed Search
    - Greedy
    - A*
    - (and many more)





Comparing Search Algorithms



Definitions

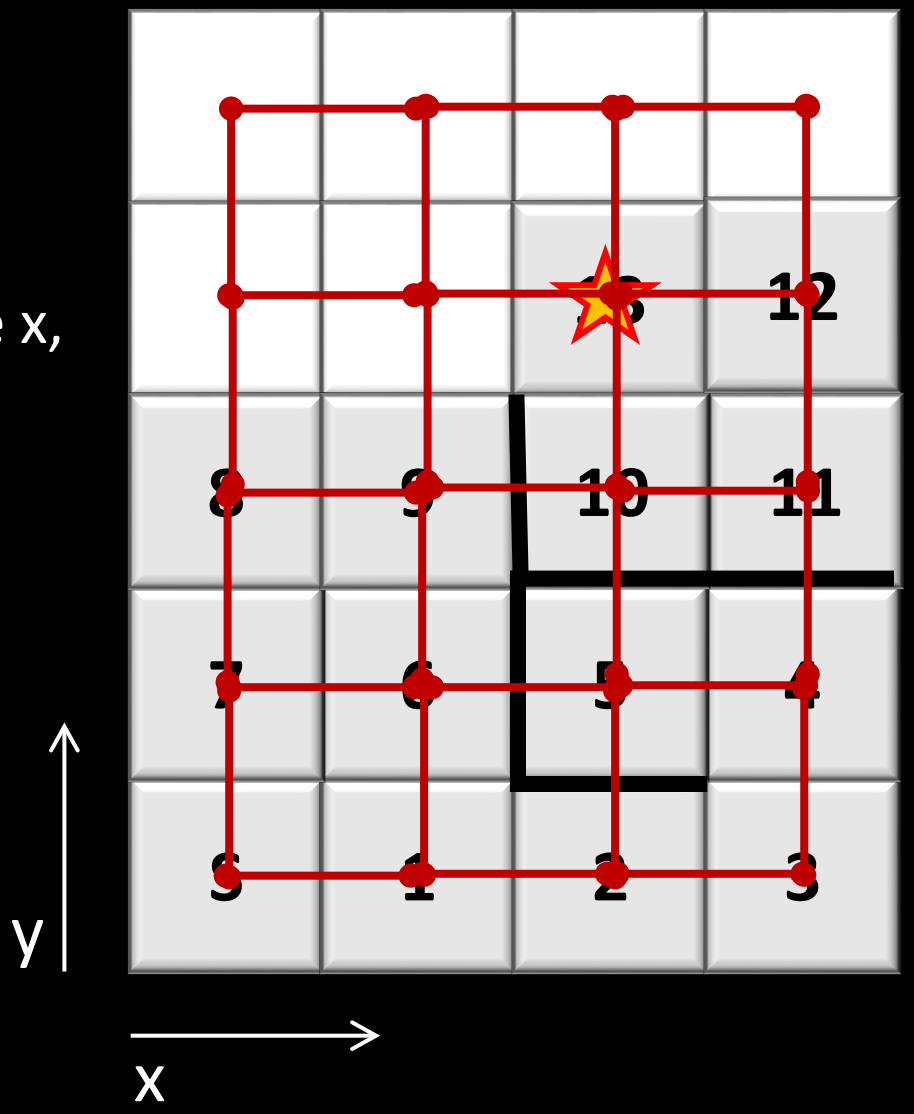
- Complete
  - A search algorithm is complete if, whenever at least one solution exists, the algorithm is guaranteed to find a solution within a finite amount of time
- Time complexity
  - The time complexity of a search algorithm is an expression for the worst-case amount of time it will take to run, expressed in terms of the maximum path length, *m,* and the maximum branching factor, *b*
- Space complexity
  - The space complexity of a search algorithm is an expression for the worst-case amount of memory that the algorithm will use, expressed in terms of *m* and *b*
- Optimality
  - A search is optimal if it is complete, and only returns cost-minimizing solutions

Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
  - **How many grid traversals will brute force take?**
    - First establish a search order
    - Advance x first, then increment y and decrease x, etc.

Search order: N, E, S, W

Find a goal

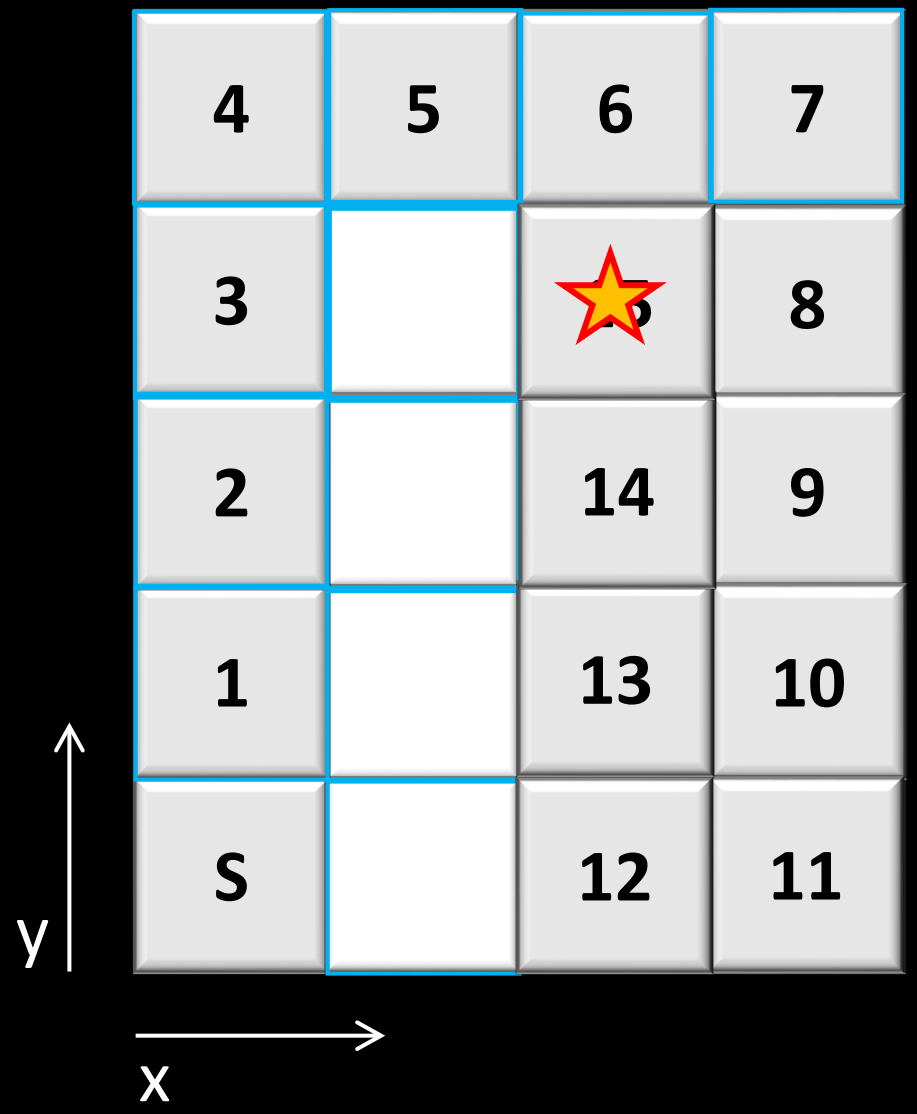


Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Depth First Search (DFS)

Search order: N, E, S, W

Find a goal

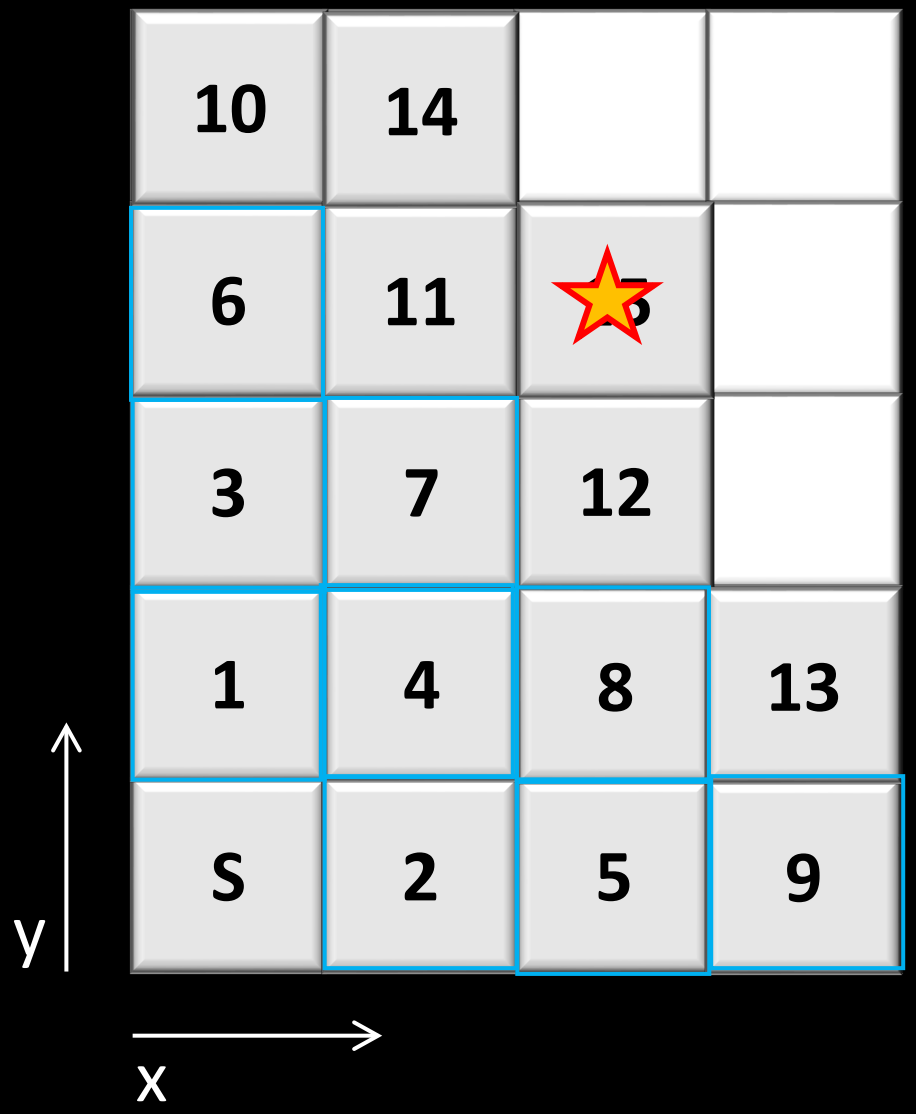


Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

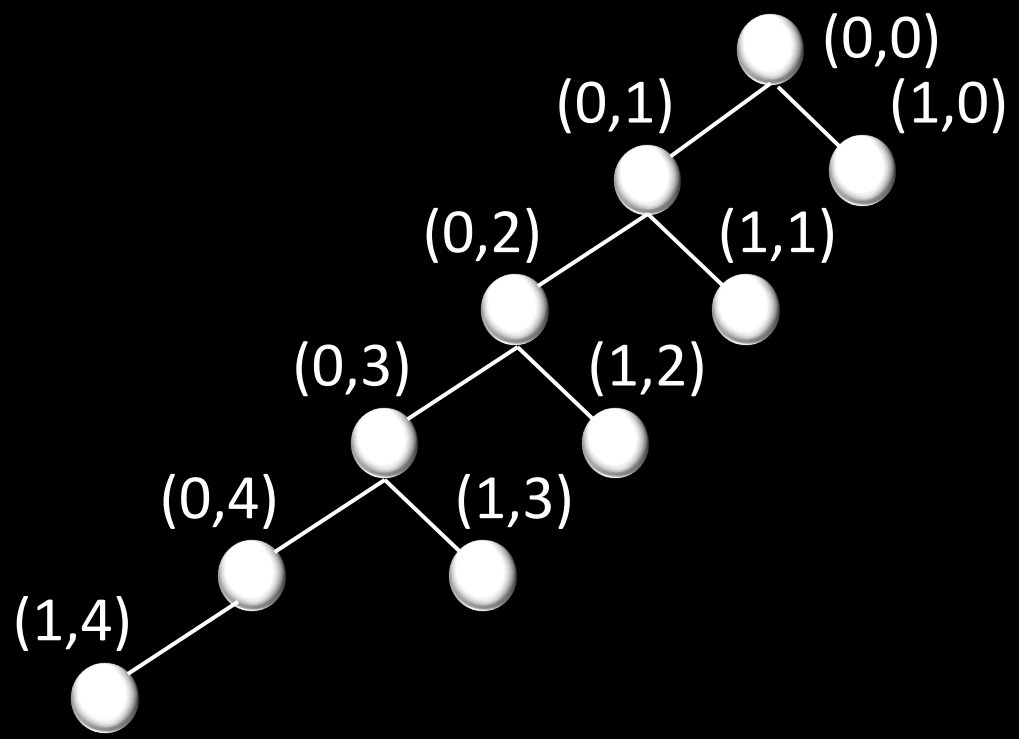
Search order: N, E, S, W

Find a goal



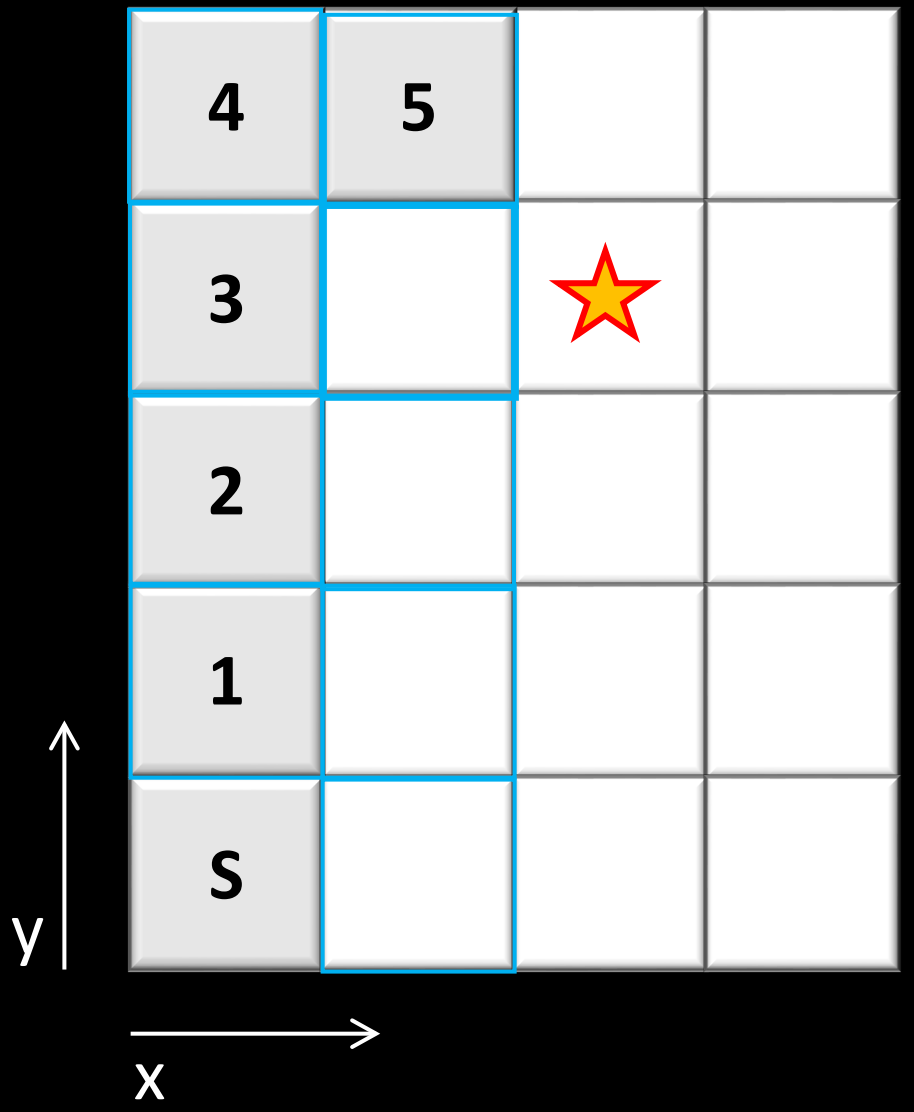
Depth First Search (DFS)

Search order: N, E, S, W



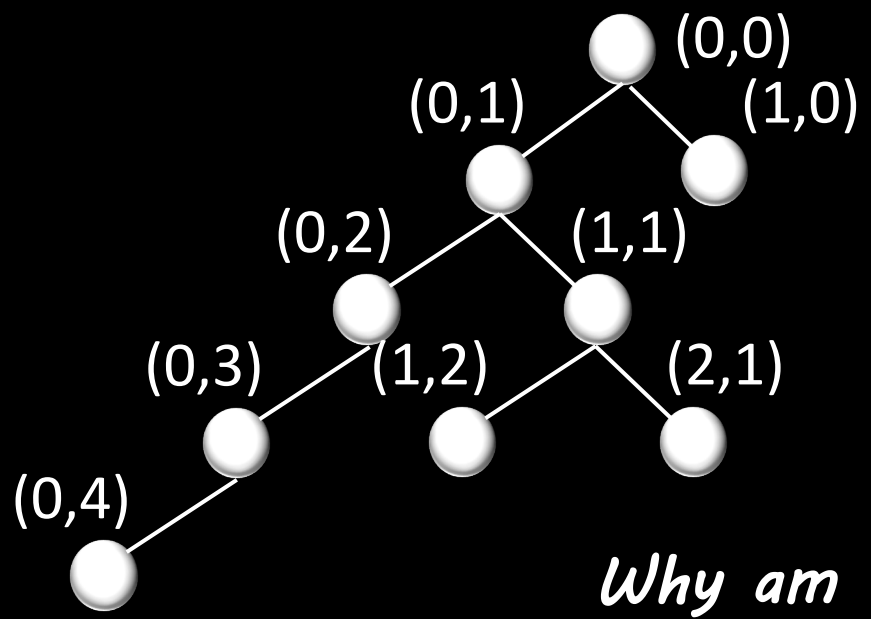
and so on…

Find a goal



Depth First Search (DFS)

Search order: N, E, S, W

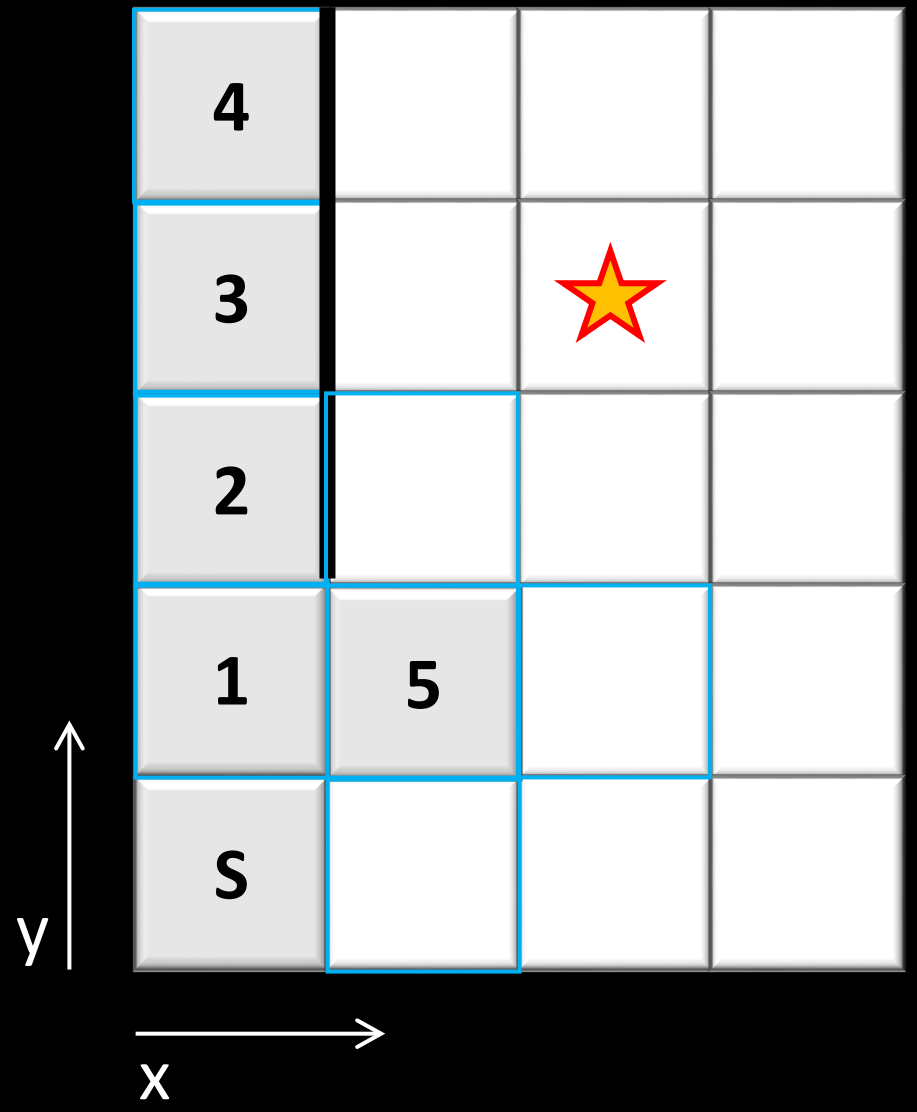


and so on…

Why am I not also adding (1,0)?

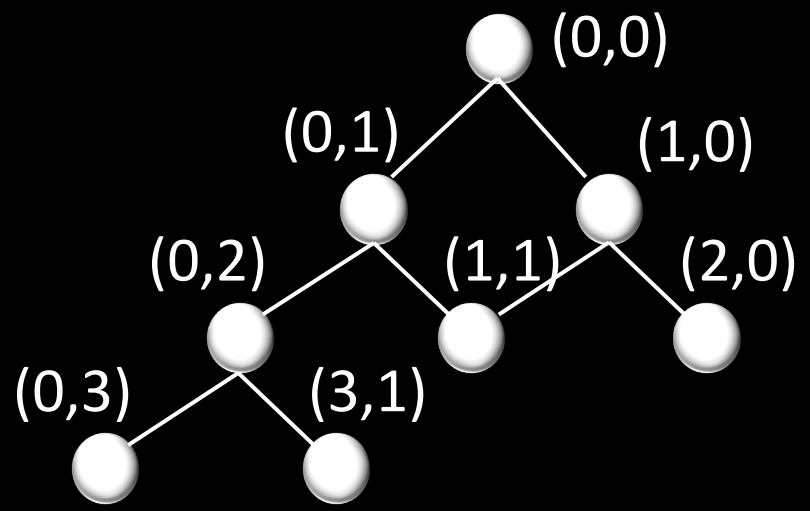
- Keep track of what is already on the frontier

Find a goal



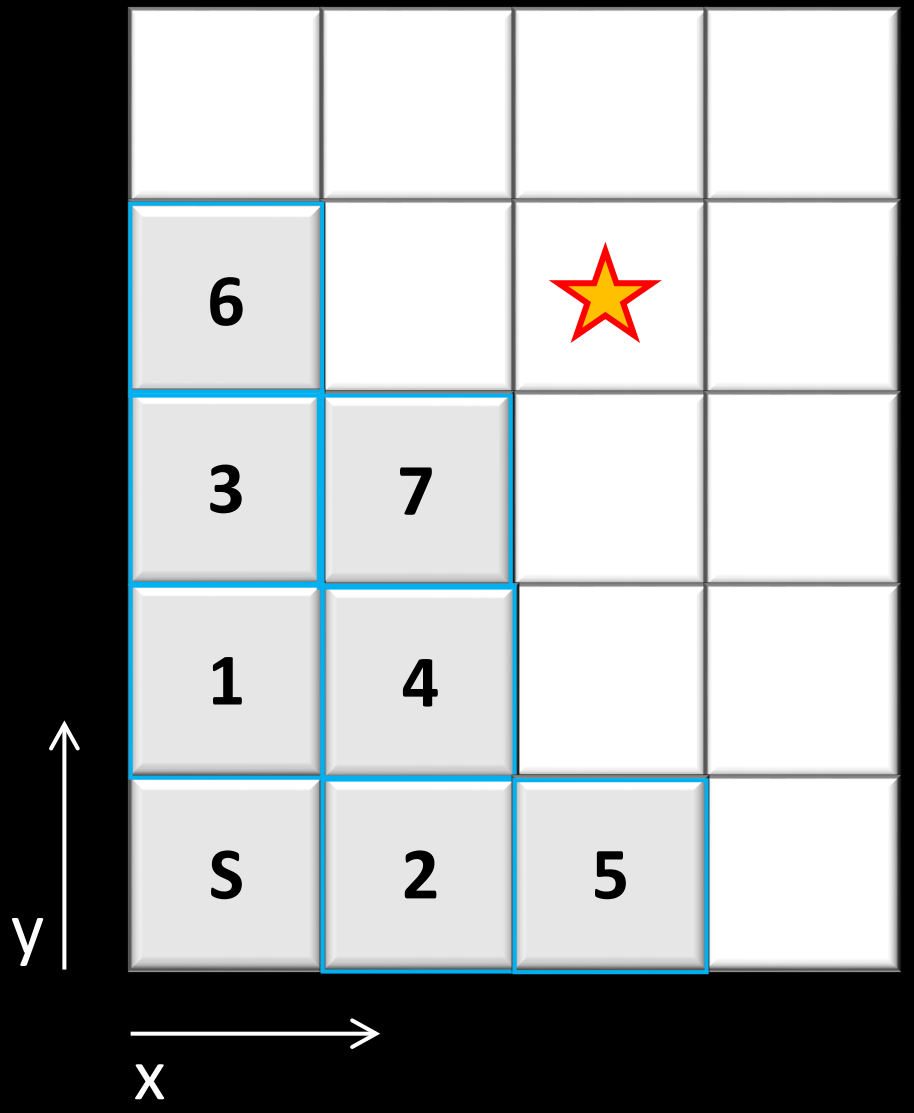
Breadth First Search (BFS)

Search order: N, E, S, W



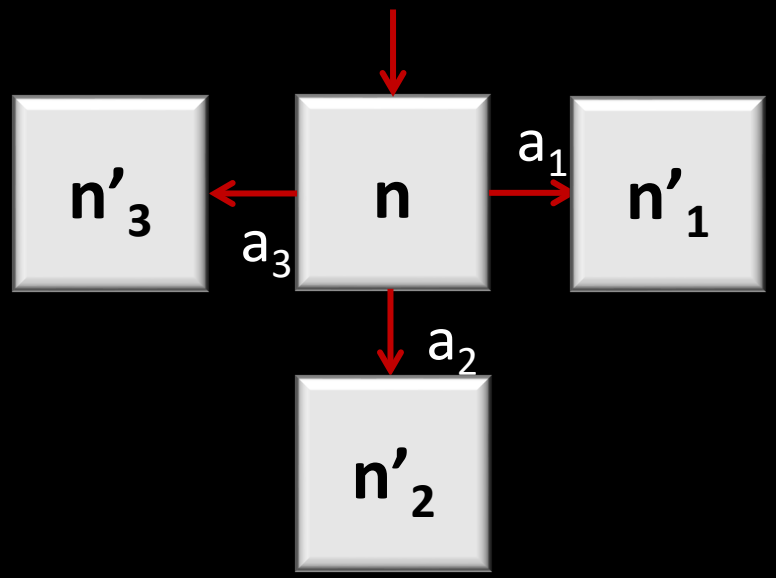
and so on…

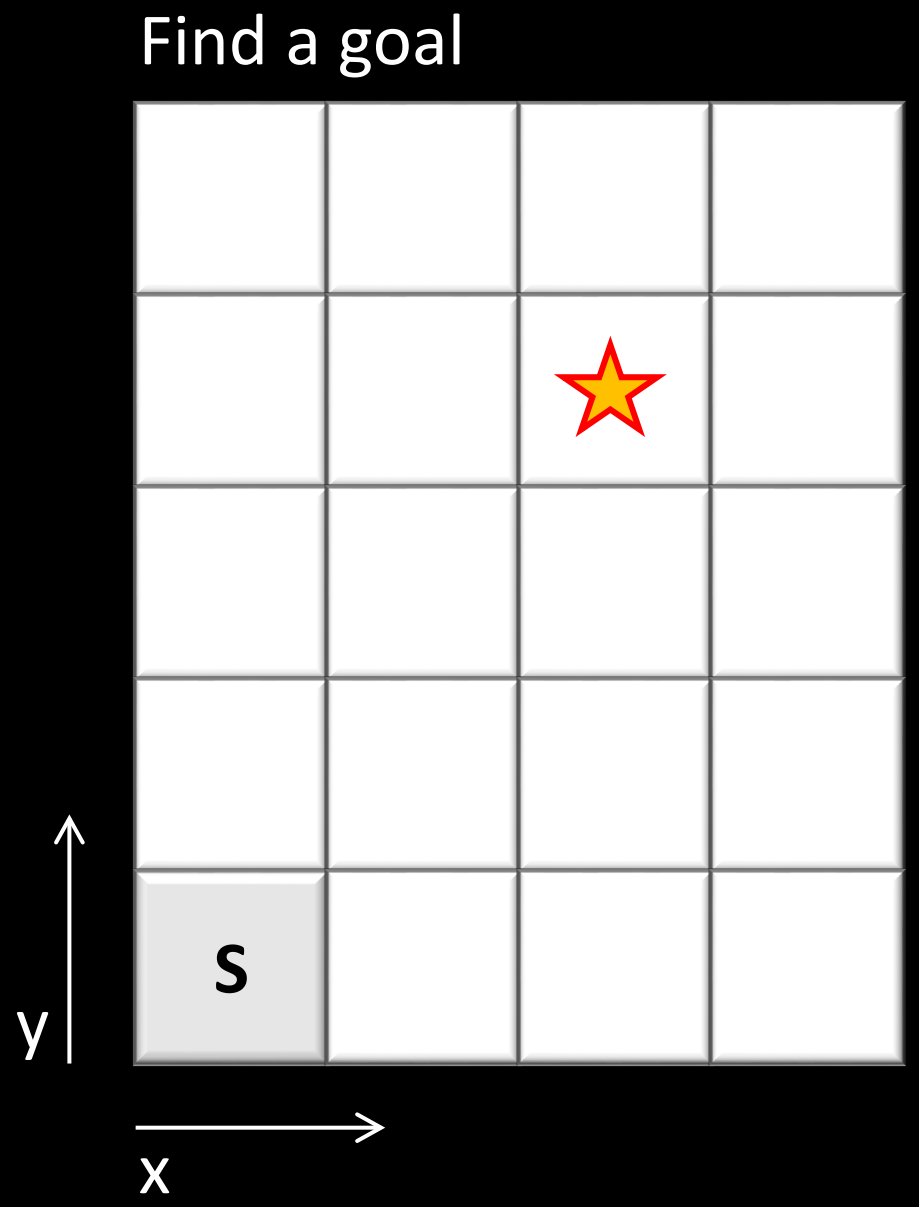
Find a goal



Search Algorithms, General

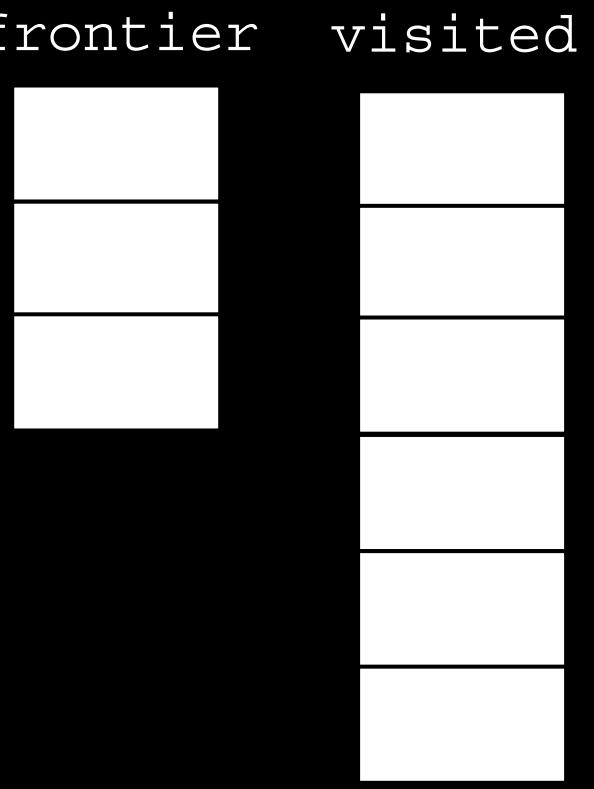
- Common graph structure
  - For every node, n
  - you have a set of actions, a
  - that moves you to a new node, n’

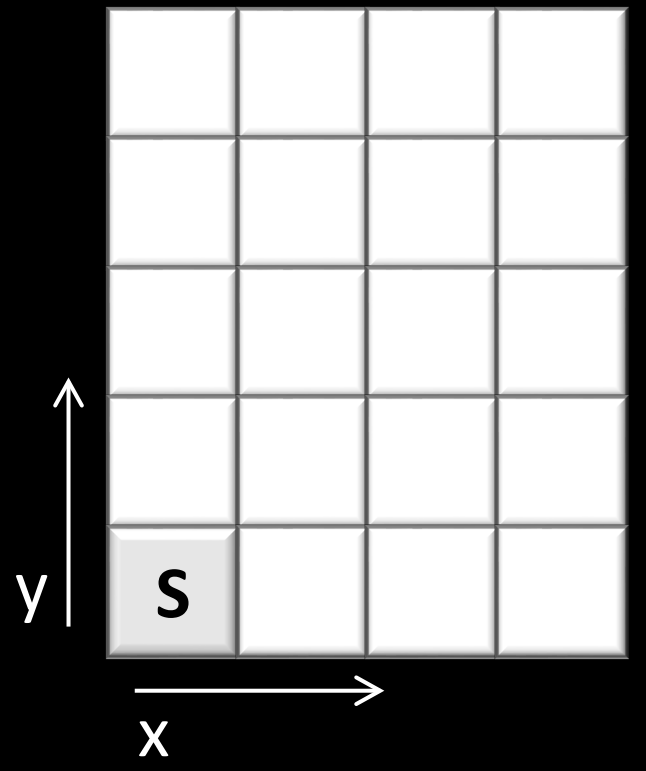




Search Algorithms, General

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

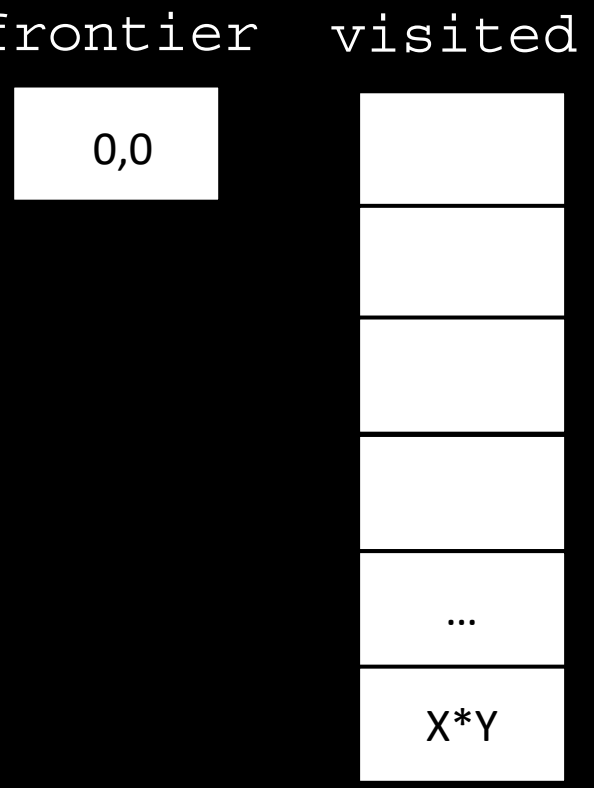


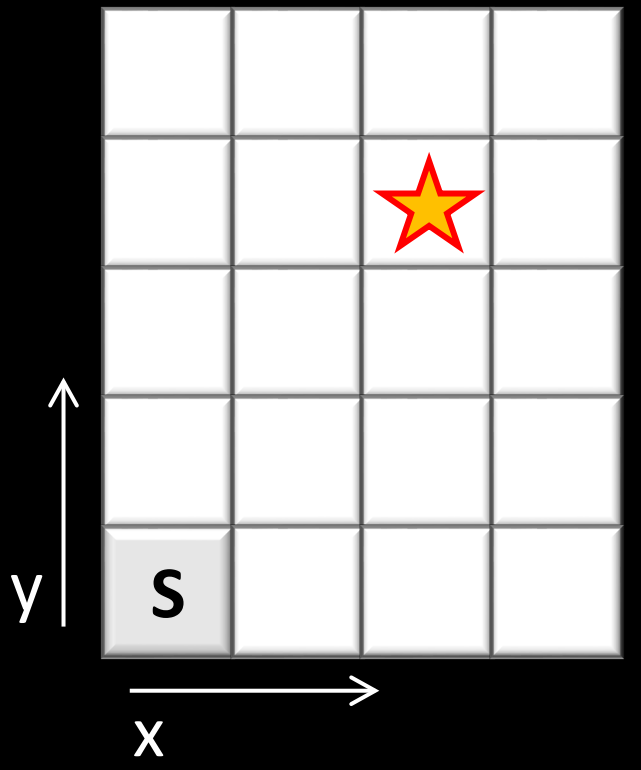


How much space to allocate to visited?

Depth First Search (DFS)

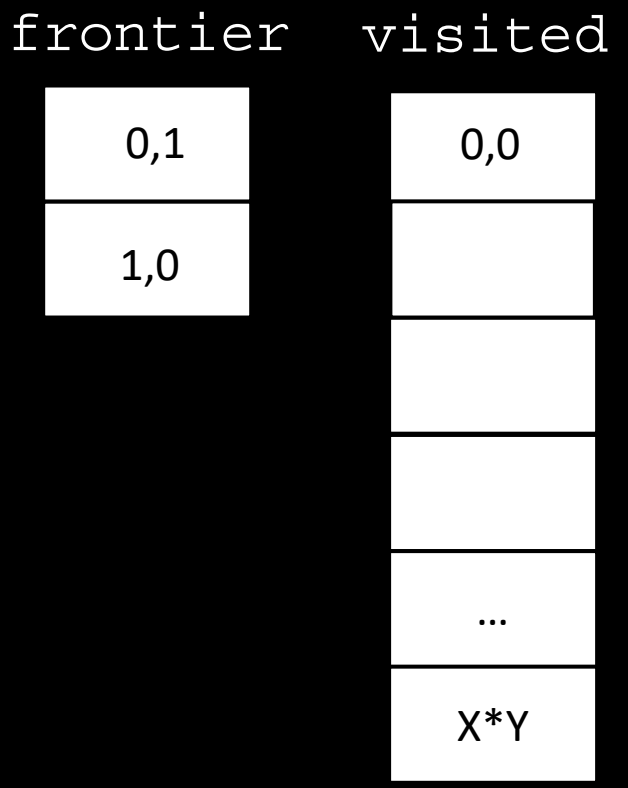
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

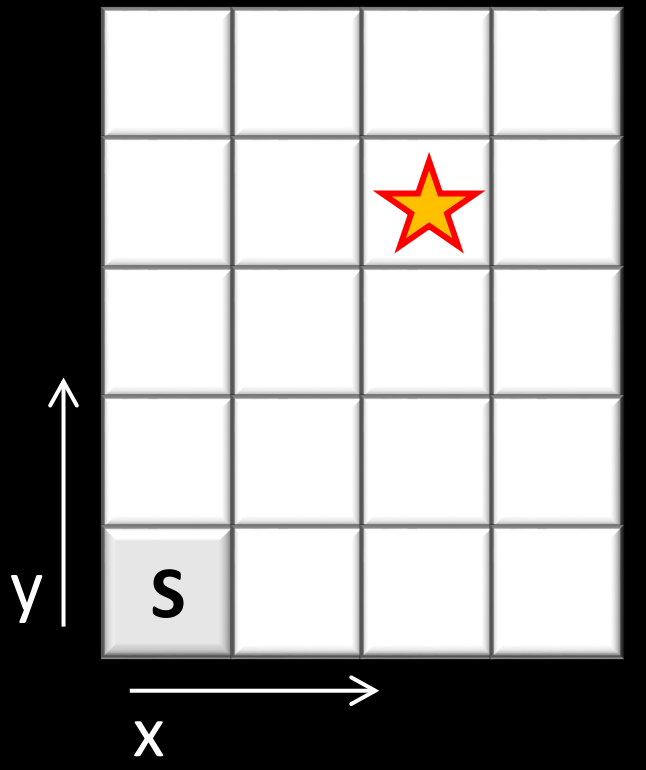




Depth First Search (DFS)

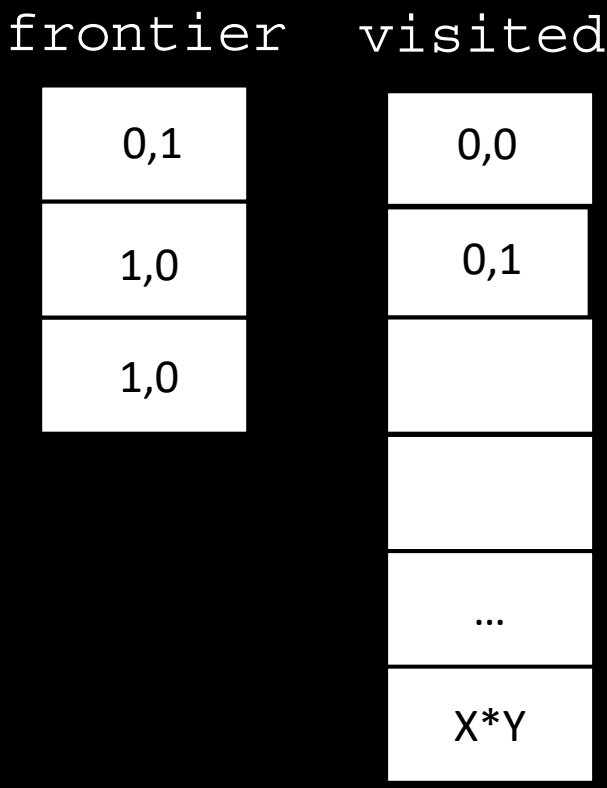
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

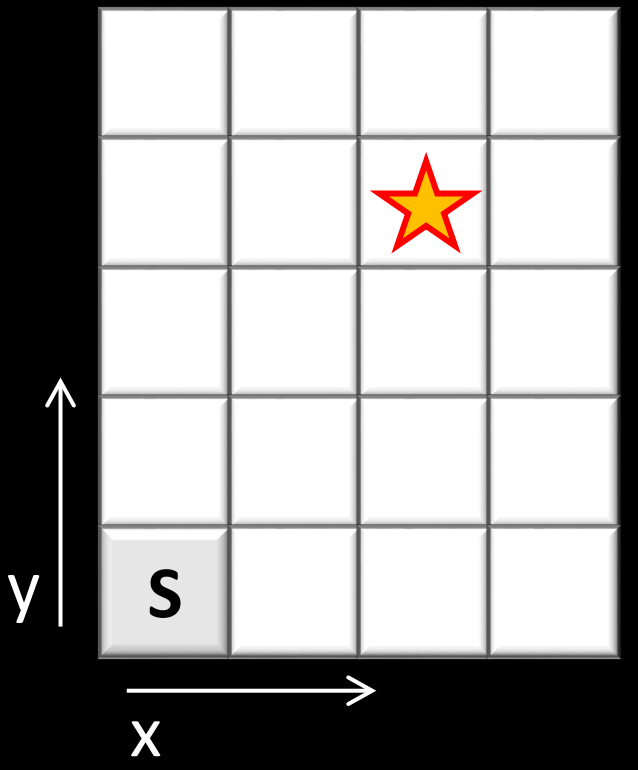




Depth First Search (DFS)

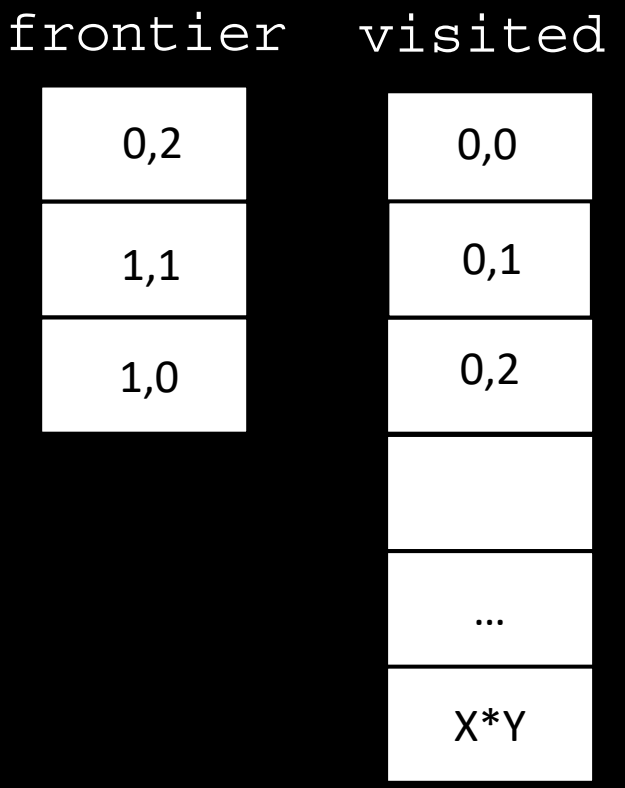
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

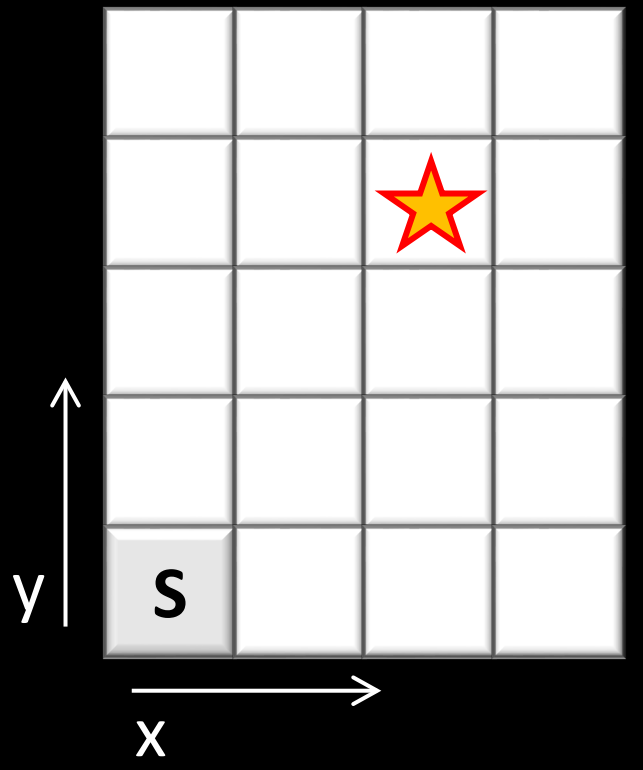




Depth First Search (DFS)

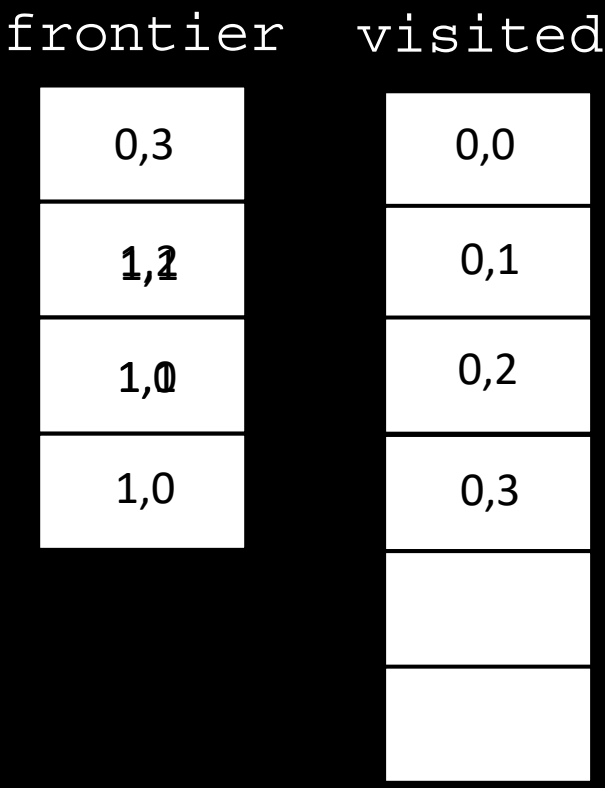
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

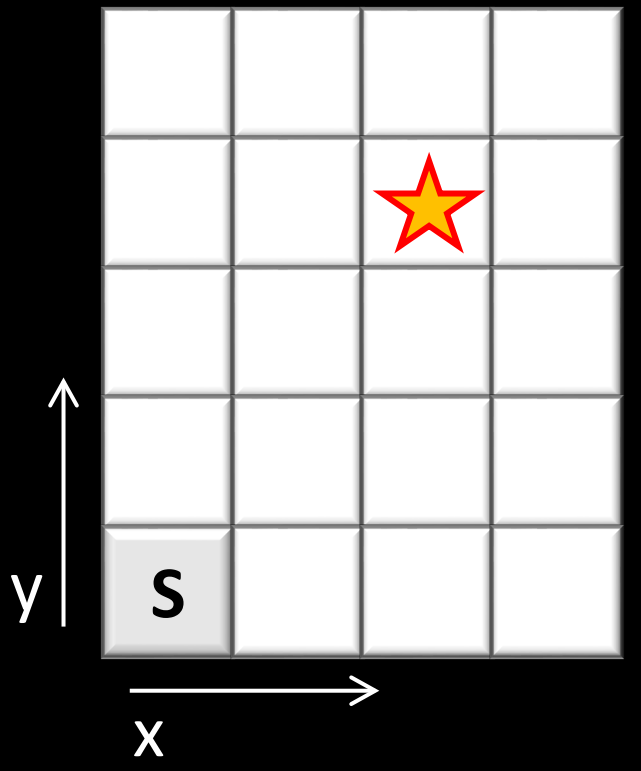




Depth First Search (DFS)

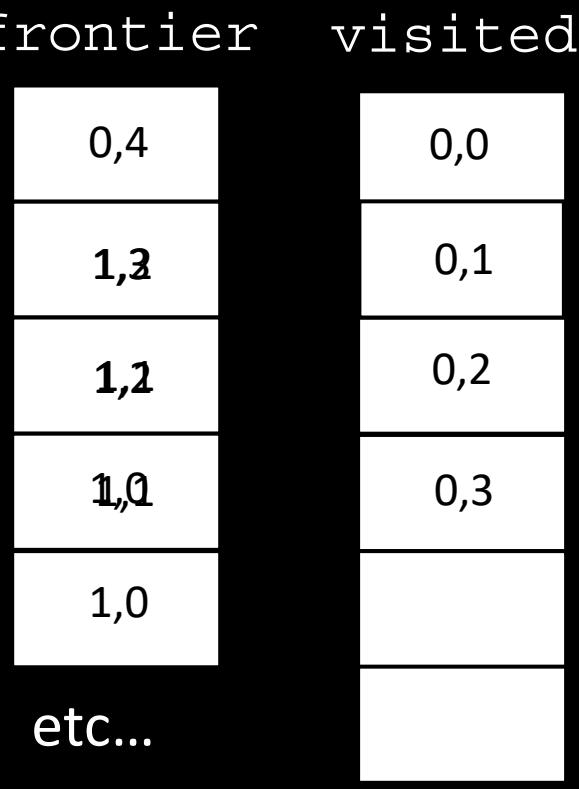
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

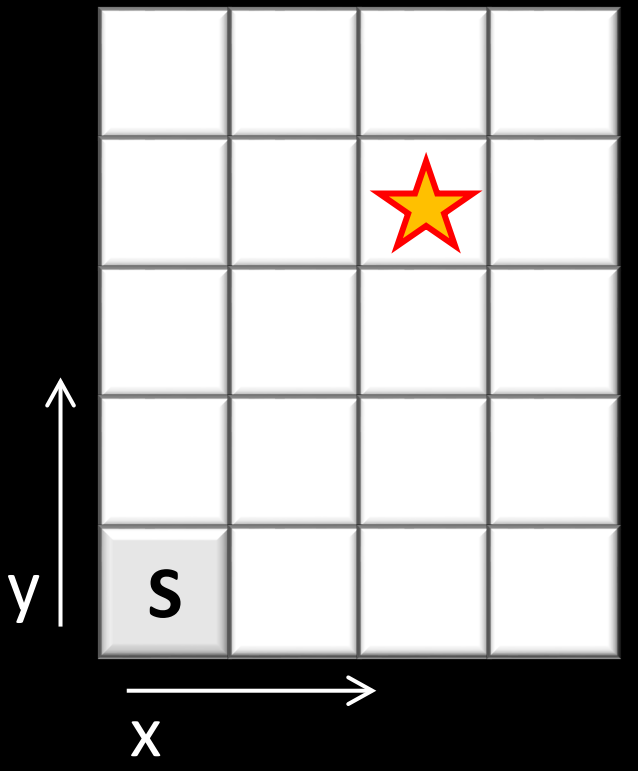




Depth First Search (DFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    append n to visited
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
            append n' to frontier
```



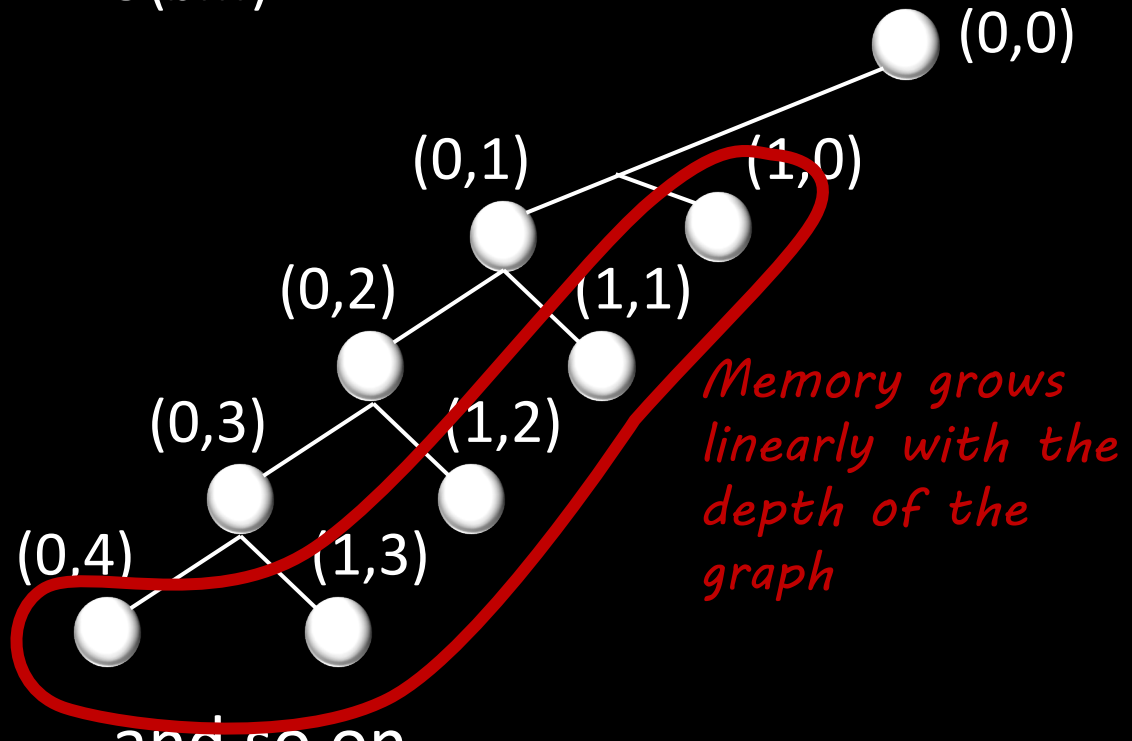


Frontier Buffer?

- Last-In First-Out (LIFO) Buffer

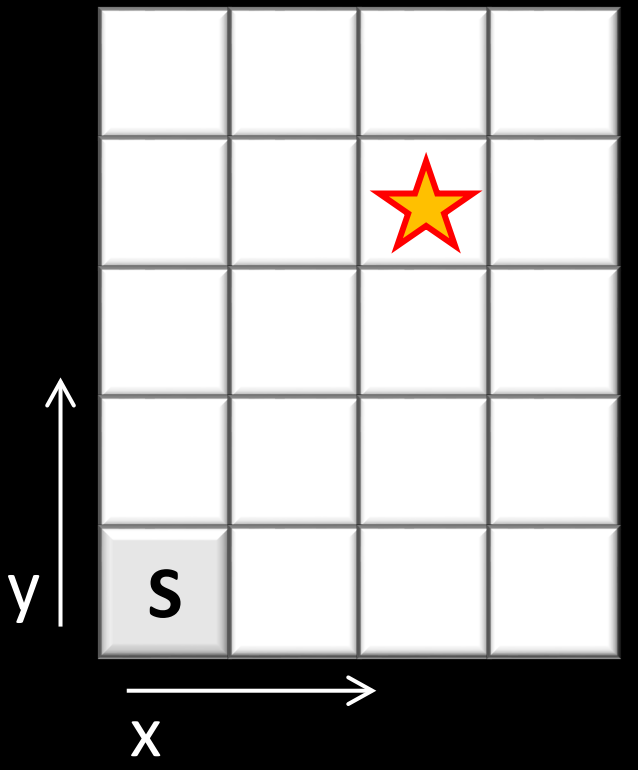
Depth First Search (DFS)

- Is it complete?
  - Yes, but only on finite graphs
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(bm)$



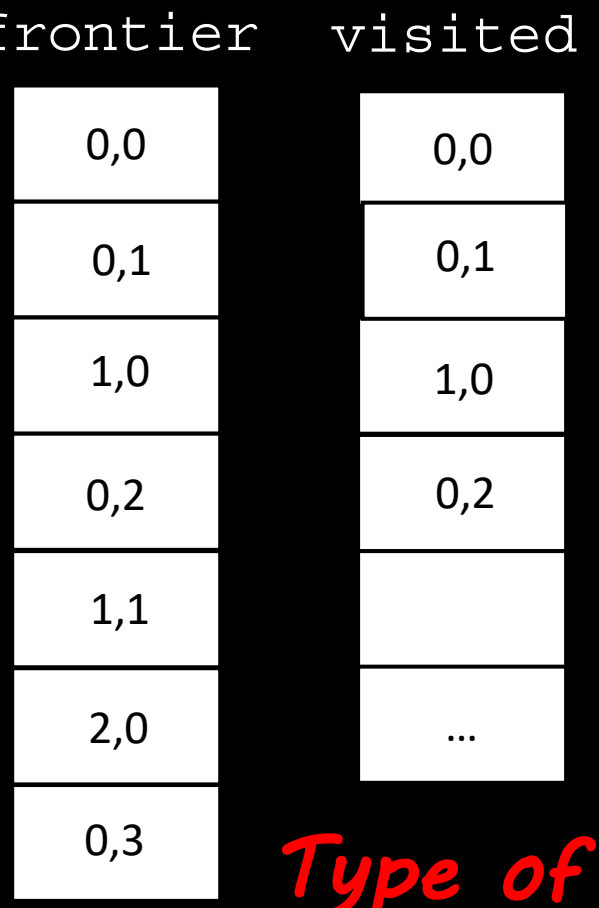
frontier
0,4
1,3
1,2
1,1
1,0

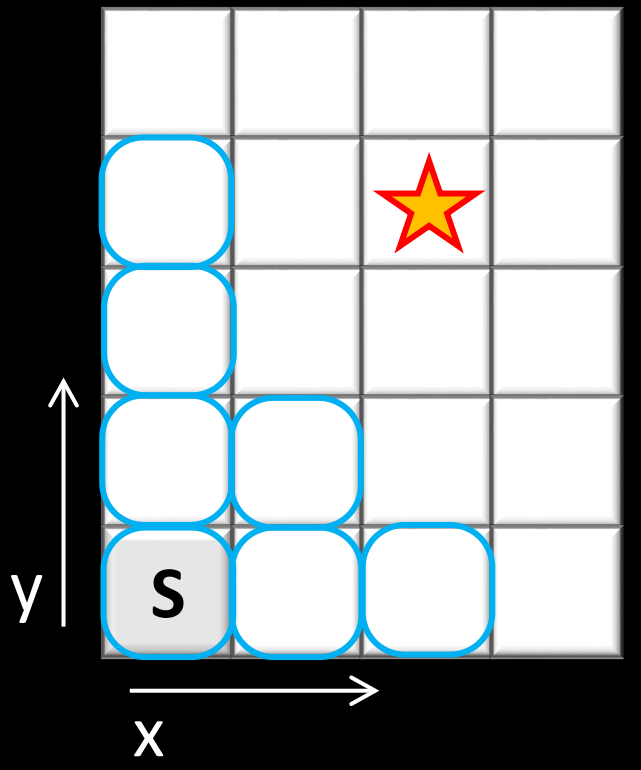
visited
0,0
0,1
0,2
0,3
…
X*Y



Breadth First Search (BFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   if n is goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
         append n' to frontier
```



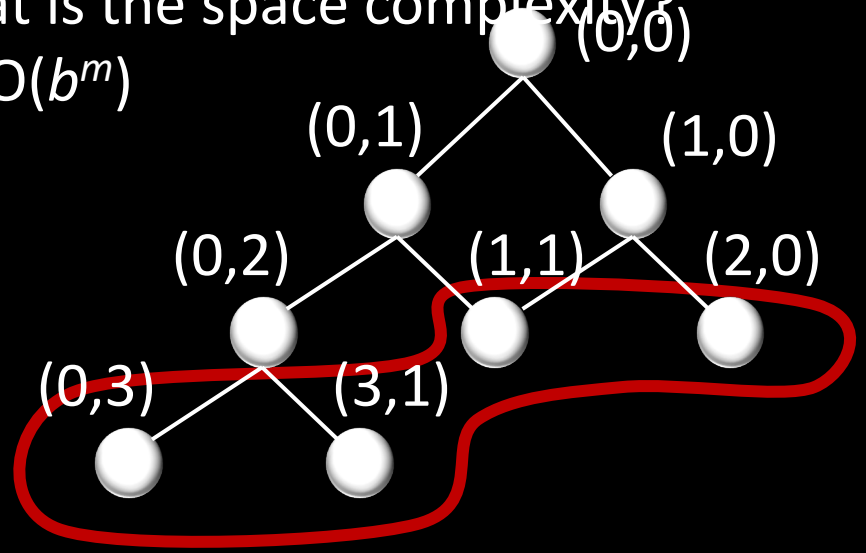


Type of Buffer?

First-In First-Out (FIFO) Buffer

Breadth First Search (BFS)

- Is it complete?
  - Yes, as long as *b* is finite
- Is it optimal?
  - Yes
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(b^m)$



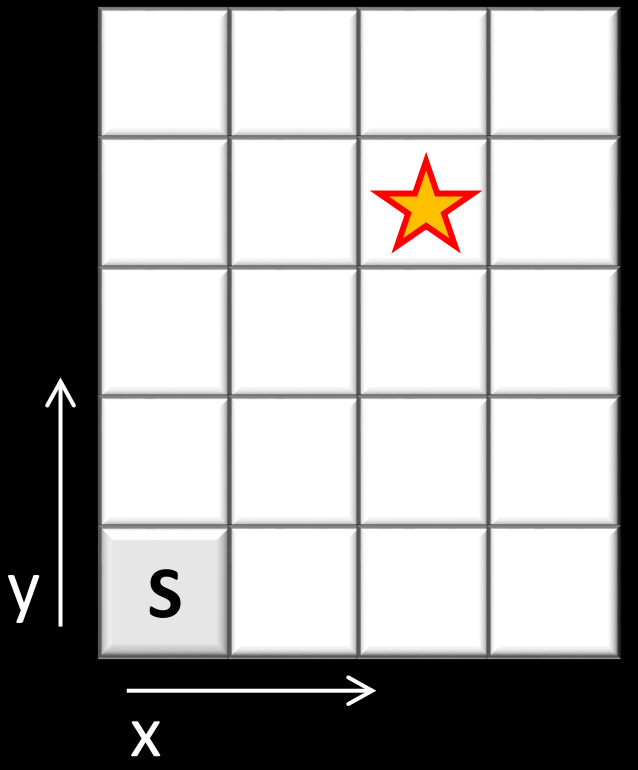
Memory grows exponentially with the depth of the graph

frontier

1,1
2,0
0,3

visited

0,0
0,1
1,0
0,2
…



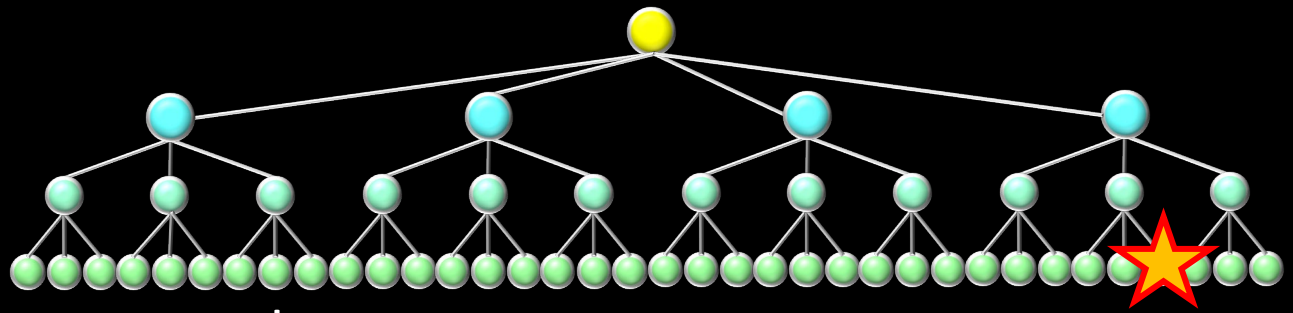
Type of Buffer?

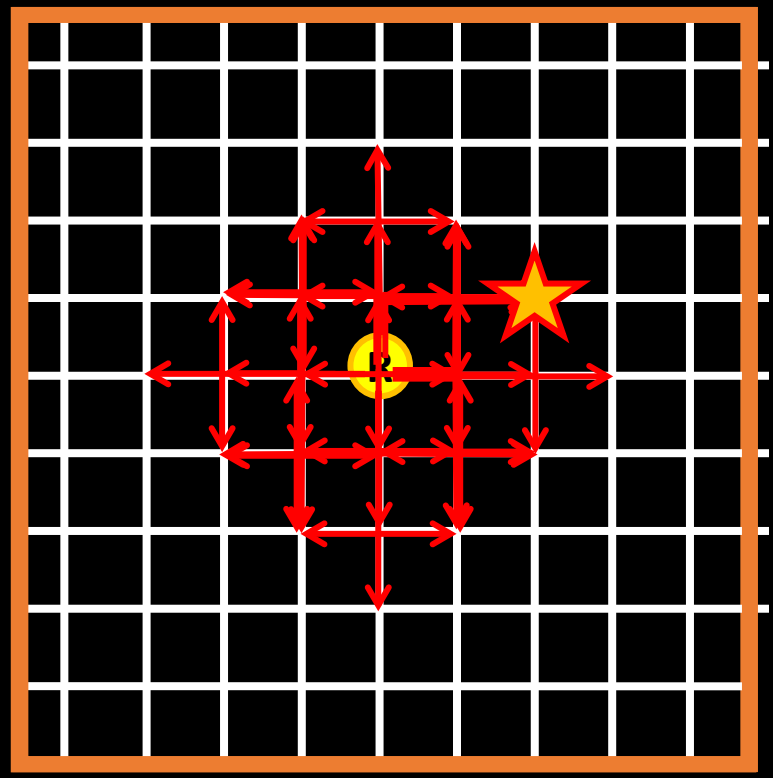
First-In First-Out (FIFO) Buffer

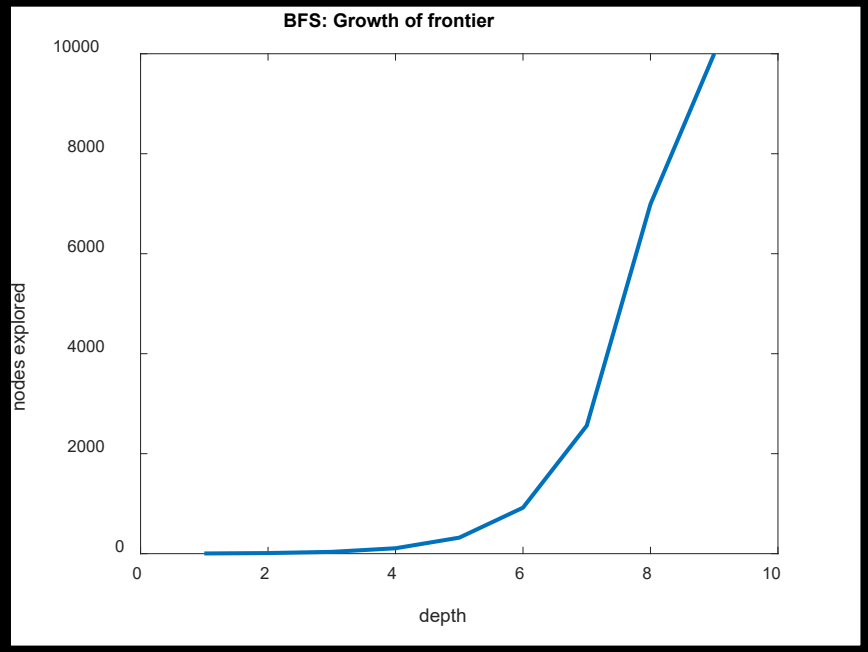
BFS: Memory and Computation

Frontier size:

- 4
- 12
- 36





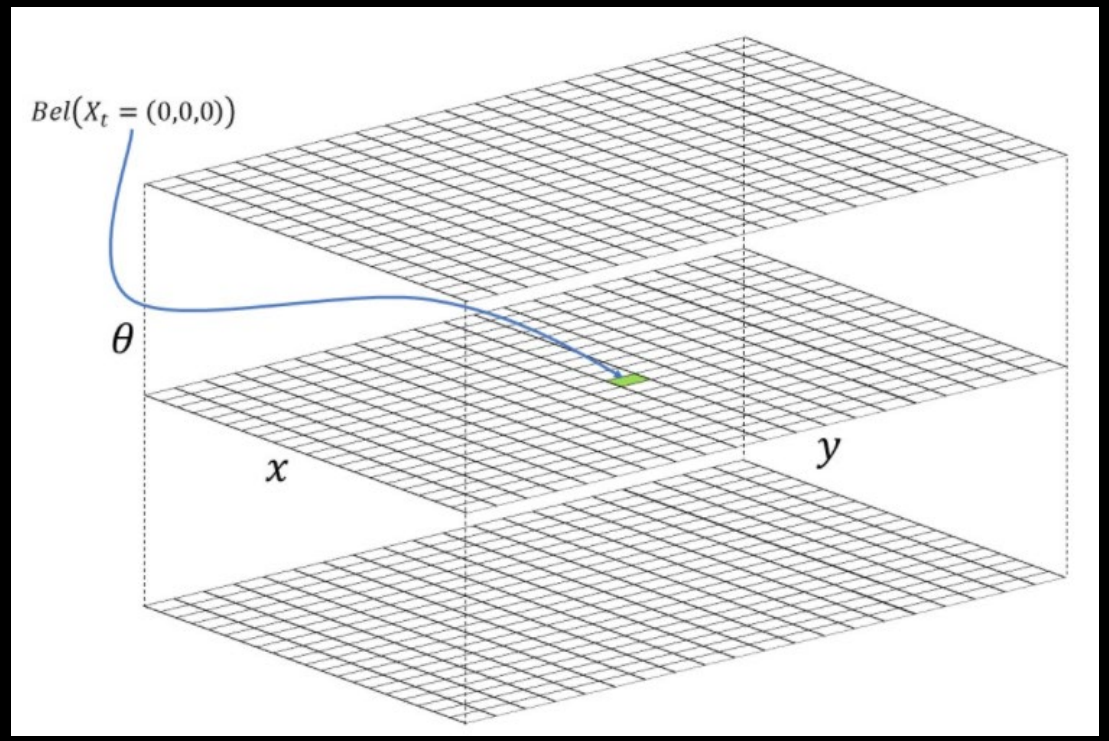


Uninformed Search Algorithms, General

- When is DFS appropriate?
  - If the memory is restricted
  - If solutions tend to occur at the same depth in the tree
- When is DFS inappropriate?
  - If some paths have infinite length / if the graph contains cycles
  - If some solutions are very deep, while others are very shallow
- When is BFS appropriate?
  - If you need to find the shortest path
  - If memory is not a problem
  - If some solutions are shallow
  - If there might be infinite paths
- When is BFS inappropriate?
  - If memory is limited / if the branching factor is very large
  - If solutions tend to be located deep in the tree

ECE4960 Robot

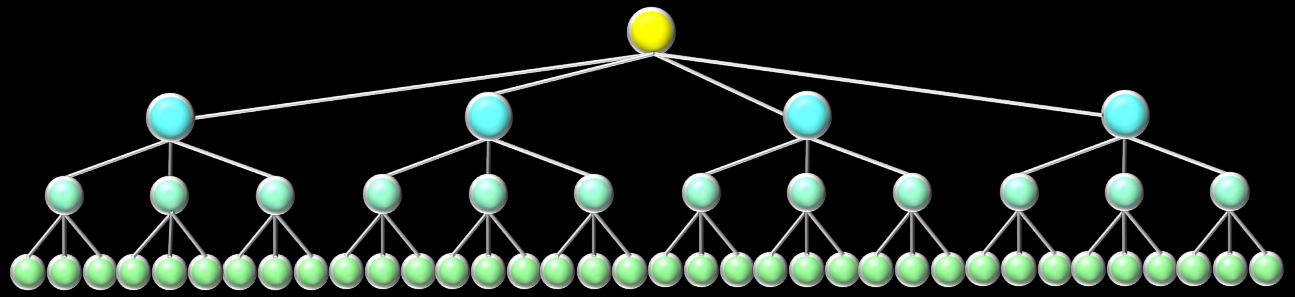
- Is BFS / DFS possible for your task on the Artemis?
  - What is the maximum branching factor?
    - b = 4
  - What is the longest path?
    - m ~ 20*20 = 400
  - Depth First Search
    - Frontier: O(*bm*) = 1,600 nodes
    - Float -> 6.4kB
    - Artemis memory?
      - 1MB flash and 384k RAM
  - Breadth First Search
    - Frontier: O(b^m) = 4^{20*20} = 6.7e240 nodes

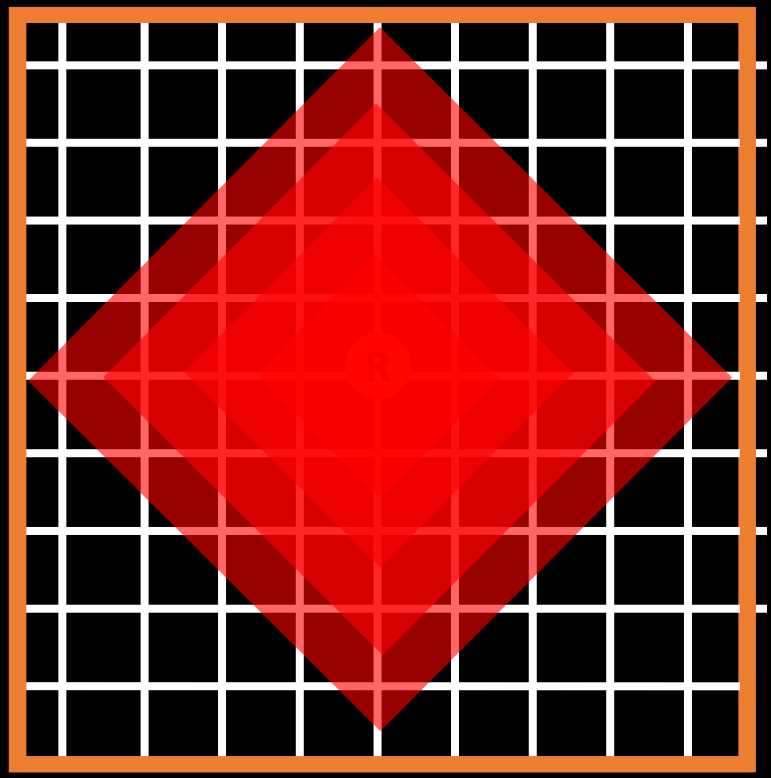


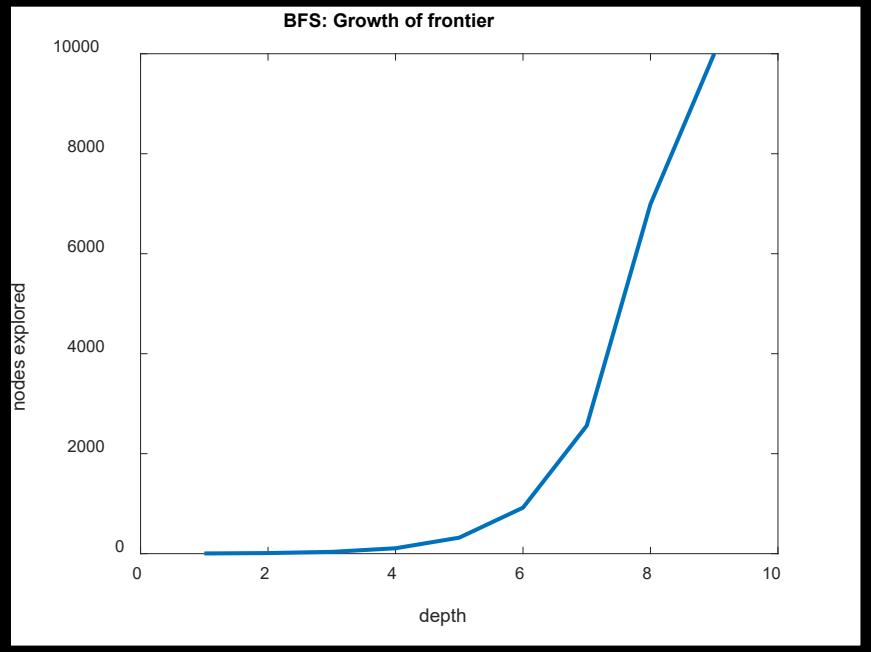
BFS: Memory and Computation

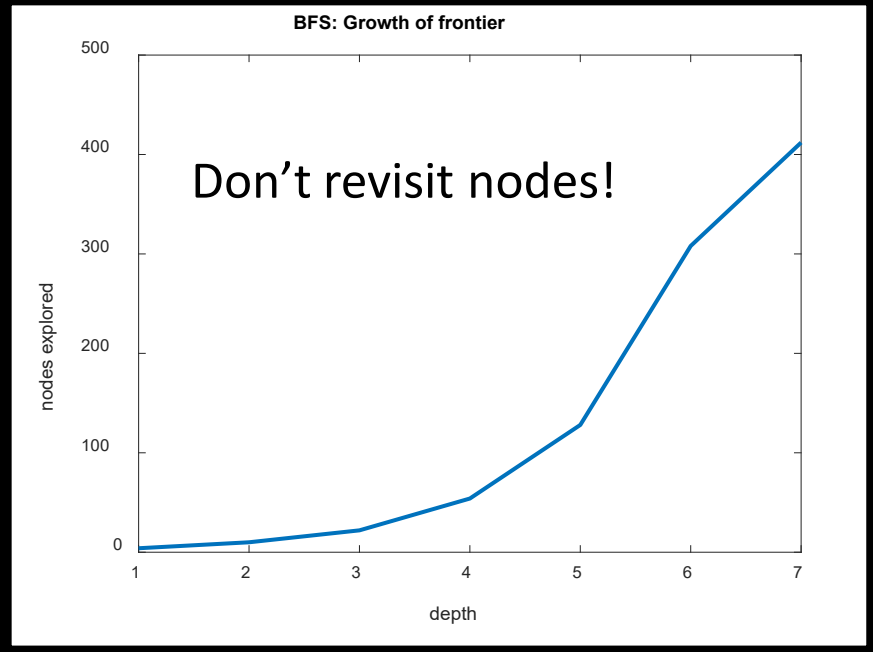
Frontier size:

- 4
- 12
- 36



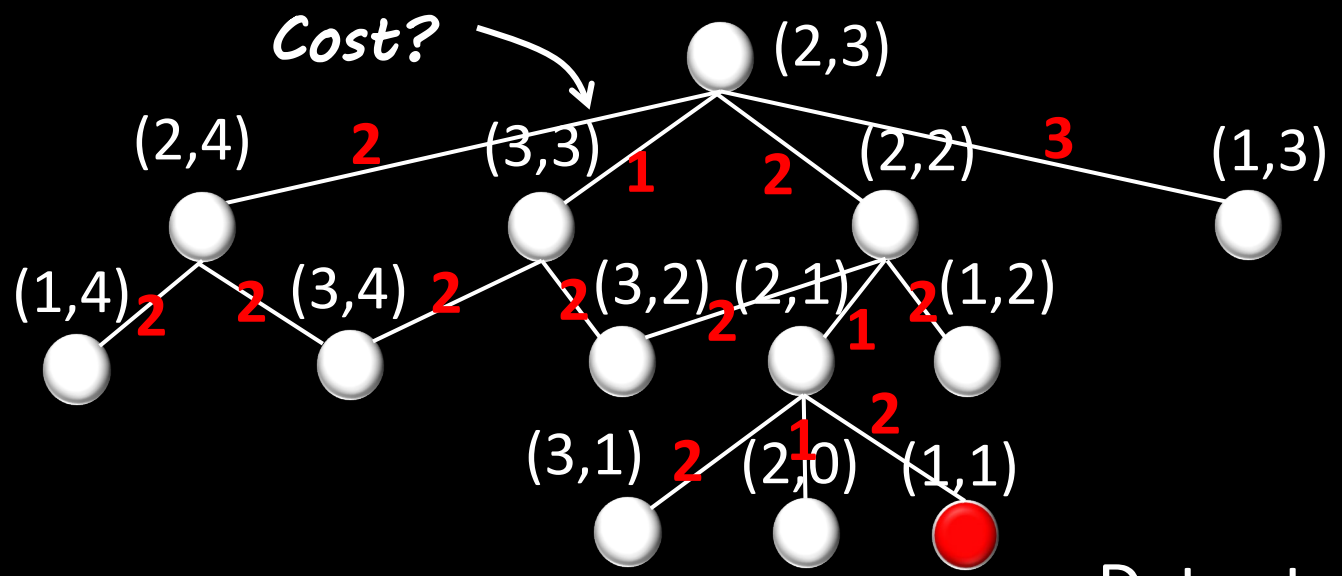






Lowest-Cost First Search

- Consider parent cost!



What node to expand next?

Data structure

- n.state
- n.parent
- n.cost
- n.action

What cost heuristic could we add?

- Go straight, cost 1
- Turn one quadrant, cost 1

	(1,4)	(2,4)	(3,4)
	(1,3)	R	(3,3)
	(1,2)	(2,2)	(3,2)
	G	(2,1)	(3,1)
		(2,0)	

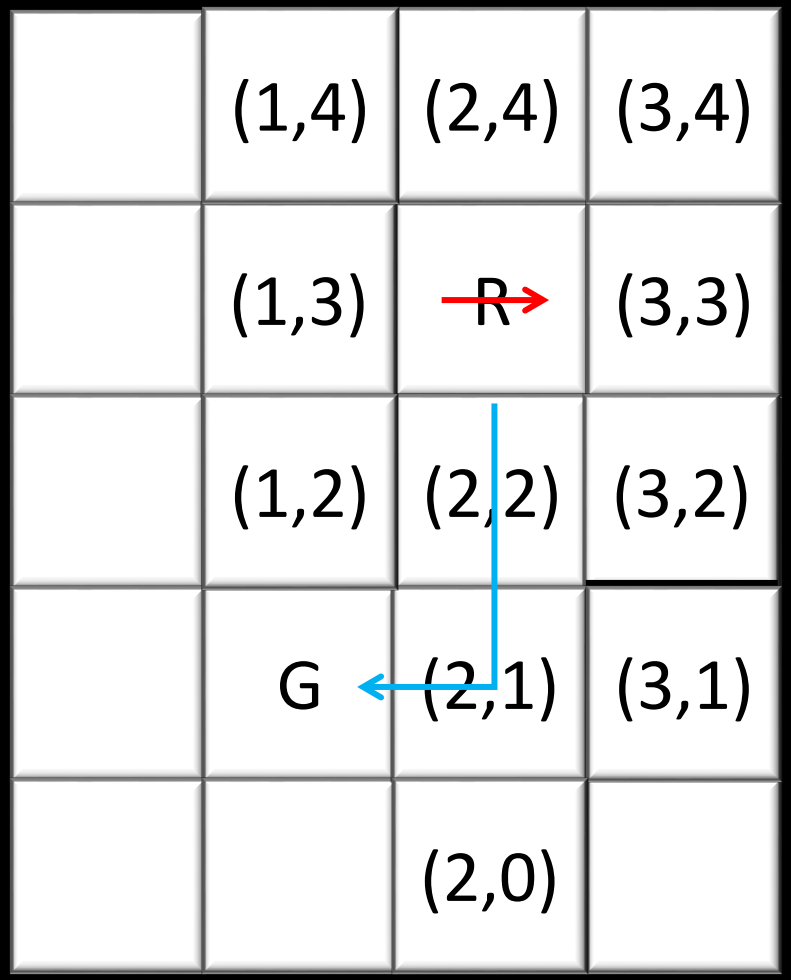
Lowest-Cost First Search

- Consider parent cost!

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    visited.append(n)
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
                priority = heuristic(goal,n')
                frontier.append(priority)
```

What cost heuristic could we add?

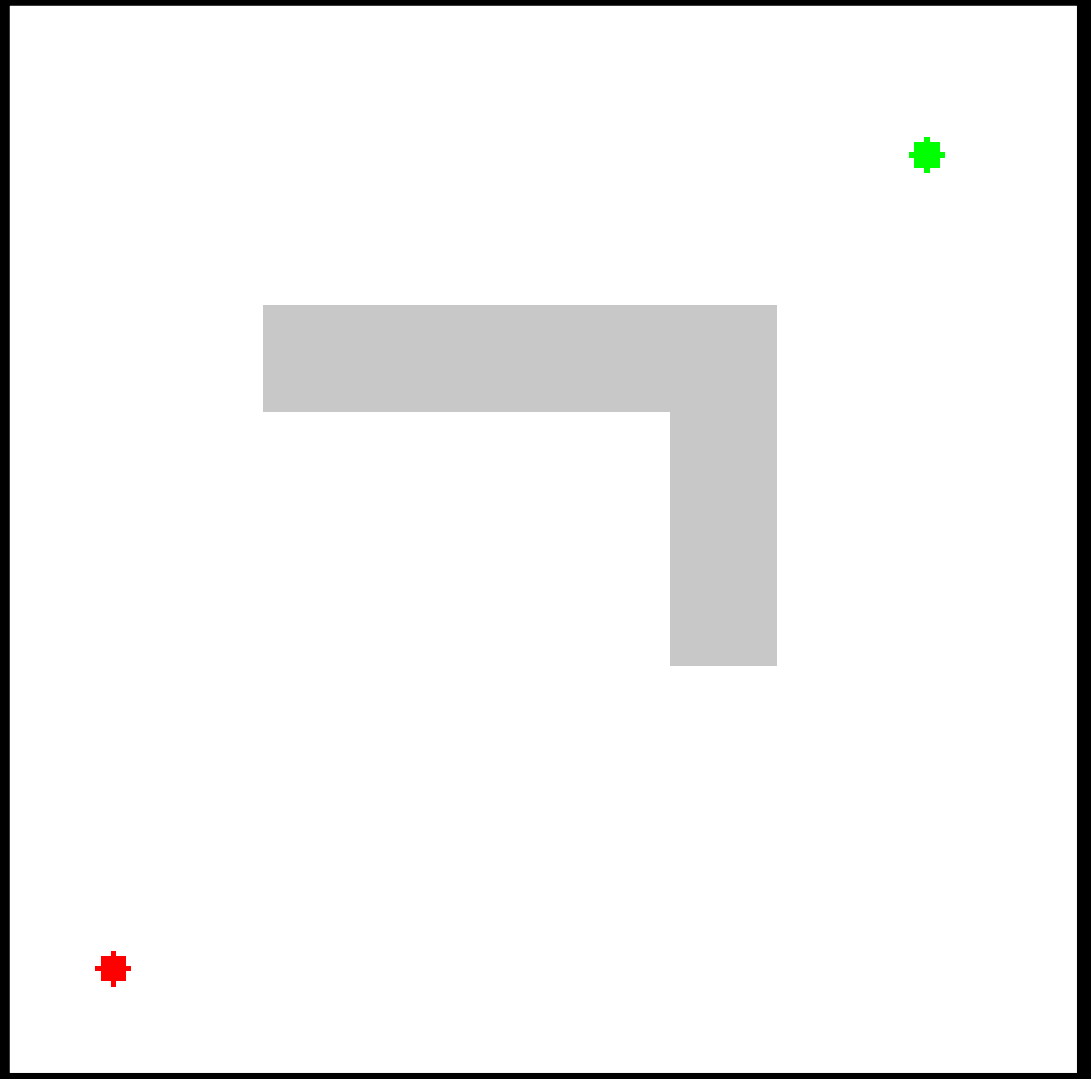
- Go straight, cost 1
- Turn one quadrant, cost 1



Lowest-Cost First Search

- Is it *complete*?
  - Yes, as long as path costs are positive
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(b^m)$

(Wikipedia)



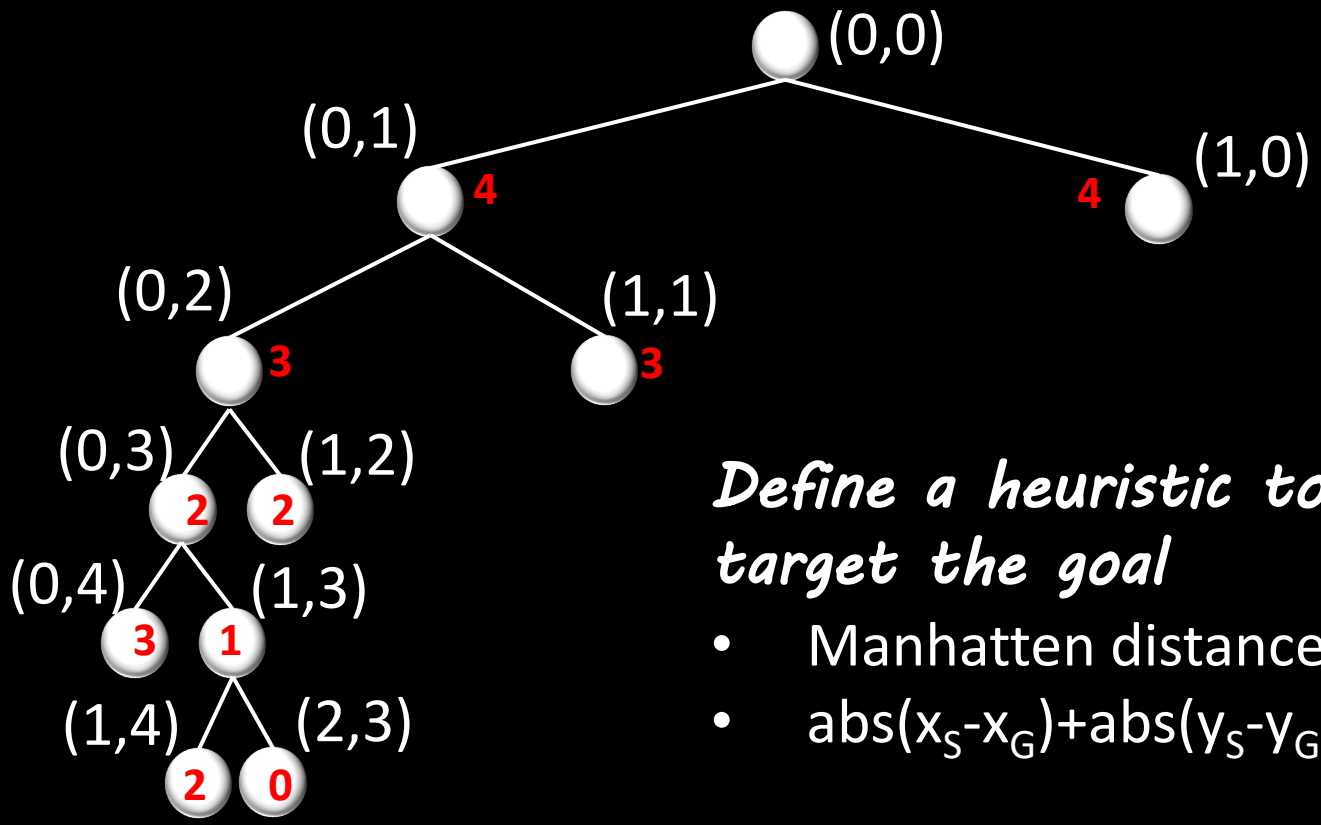
Could we be smarter?

- Sure, *you know the graph and you know the goal is*!
- …Informed search
  - Consider parent cost, and..
  - ..estimate the shortest path to the “goal”
- Assign a value to the frontier
  - Pick frontier closest to the goal (minimize distance)

Informed Search

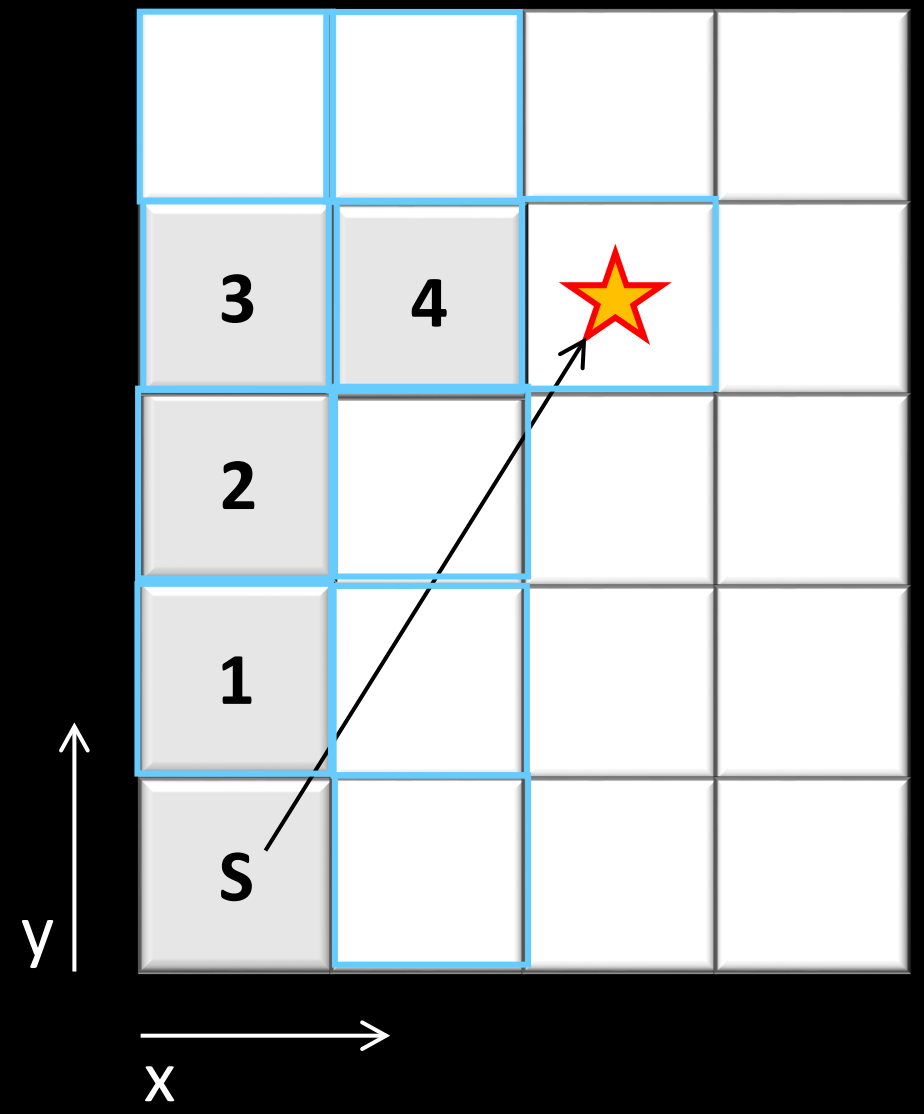
Search order: N, E, S, W

- Greedy Search



Define a heuristic to target the goal

- Manhatten distance
- $abs(x_S-x_G)+abs(y_S-y_G)$

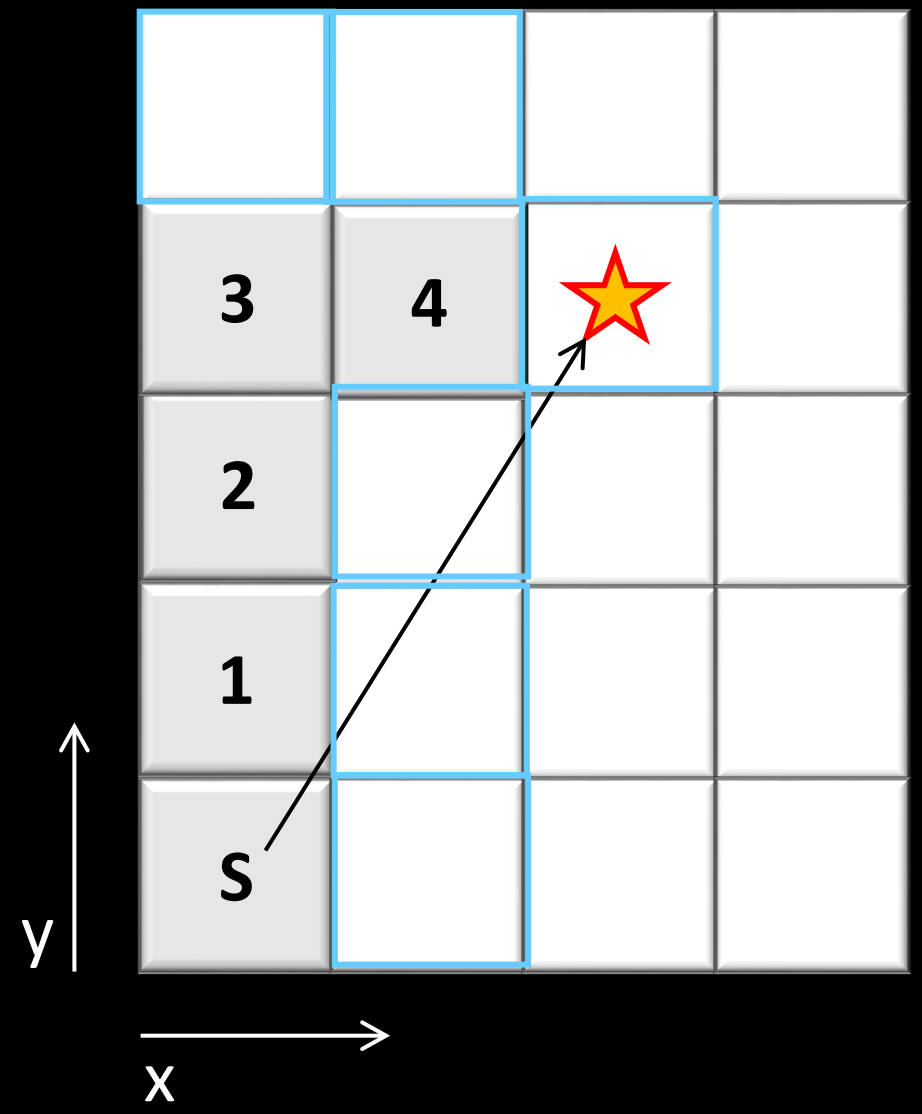


Informed Search

Search order: N, E, S, W

- Greedy Search

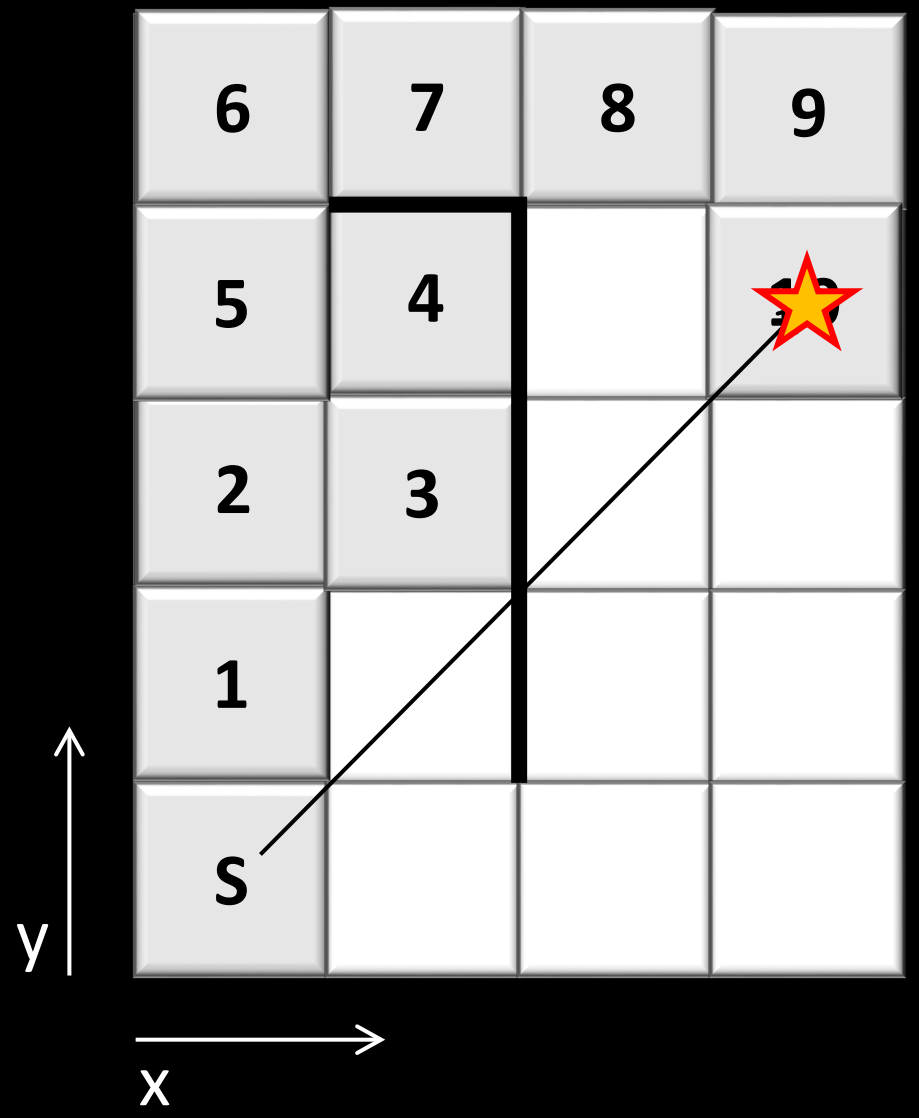
```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    visited.append(n)
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
            priority = heuristic(goal,n')
            frontier.append(priority)
```



Informed Search

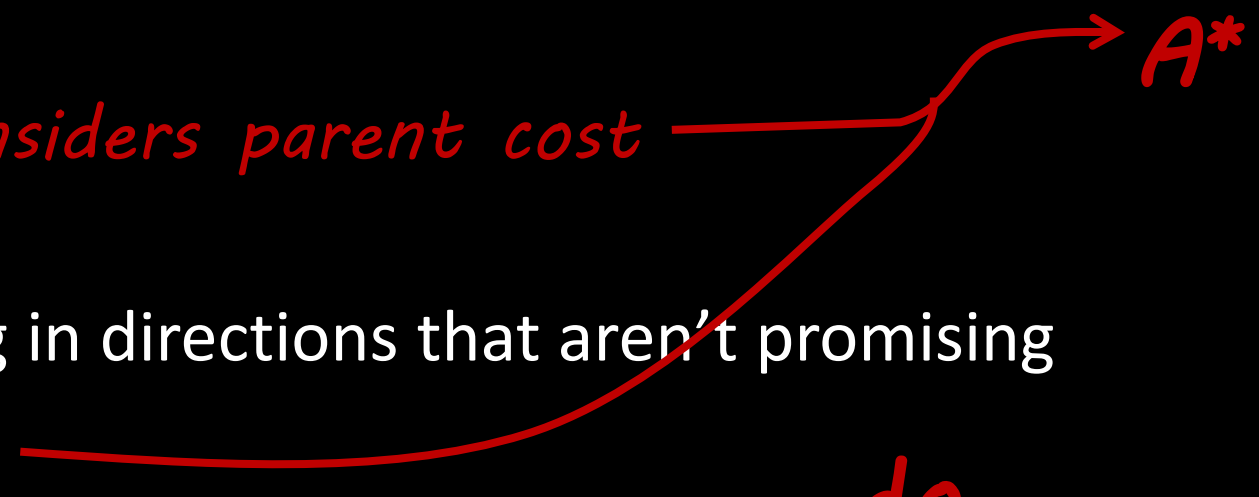
Search order: N, E, S, W

- Greedy Search
  - Complete?
    - No
  - Time complexity?
    - $O(b^m)$
  - Space complexity?
    - $O(b^m)$
  - Optimal?
    - no…



Search Algorithms, General

- Breadth First Search
  - *Complete* and optimal
  - …but searches *everything*
- Lowest-Cost First Algorithm
  - *Complete and optimal*
  - …but it wastes time exploring in directions that aren't promising
- Greedy Search
  - *Complete (in most cases)*
  - …only explores promising directions



Can we do better?

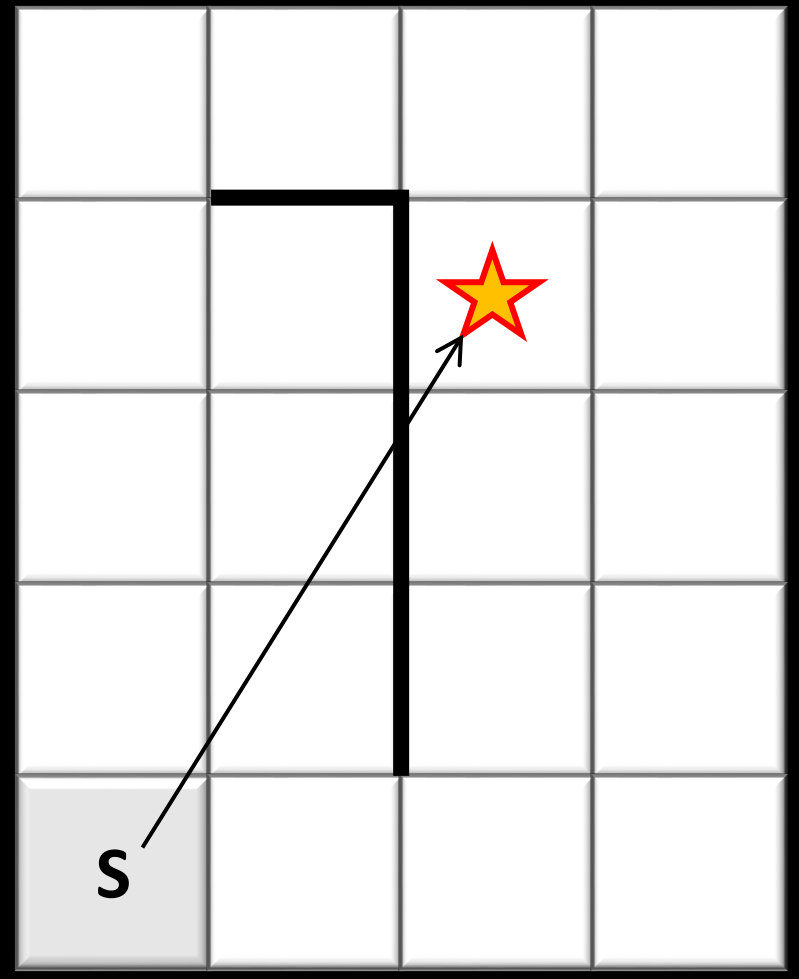
Informed Search

Search order: N, E, S, W

- A* (“A-star”)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    if n = goal, return solution
    for all actions in n
        n’ = a(n)
        if ((n’ not visited or
            (visited and n’.cost < n_old.cost))
                priority = heuristic(goal,n’)+cost
                frontier.append(priority)
                visited.append(n’)
```

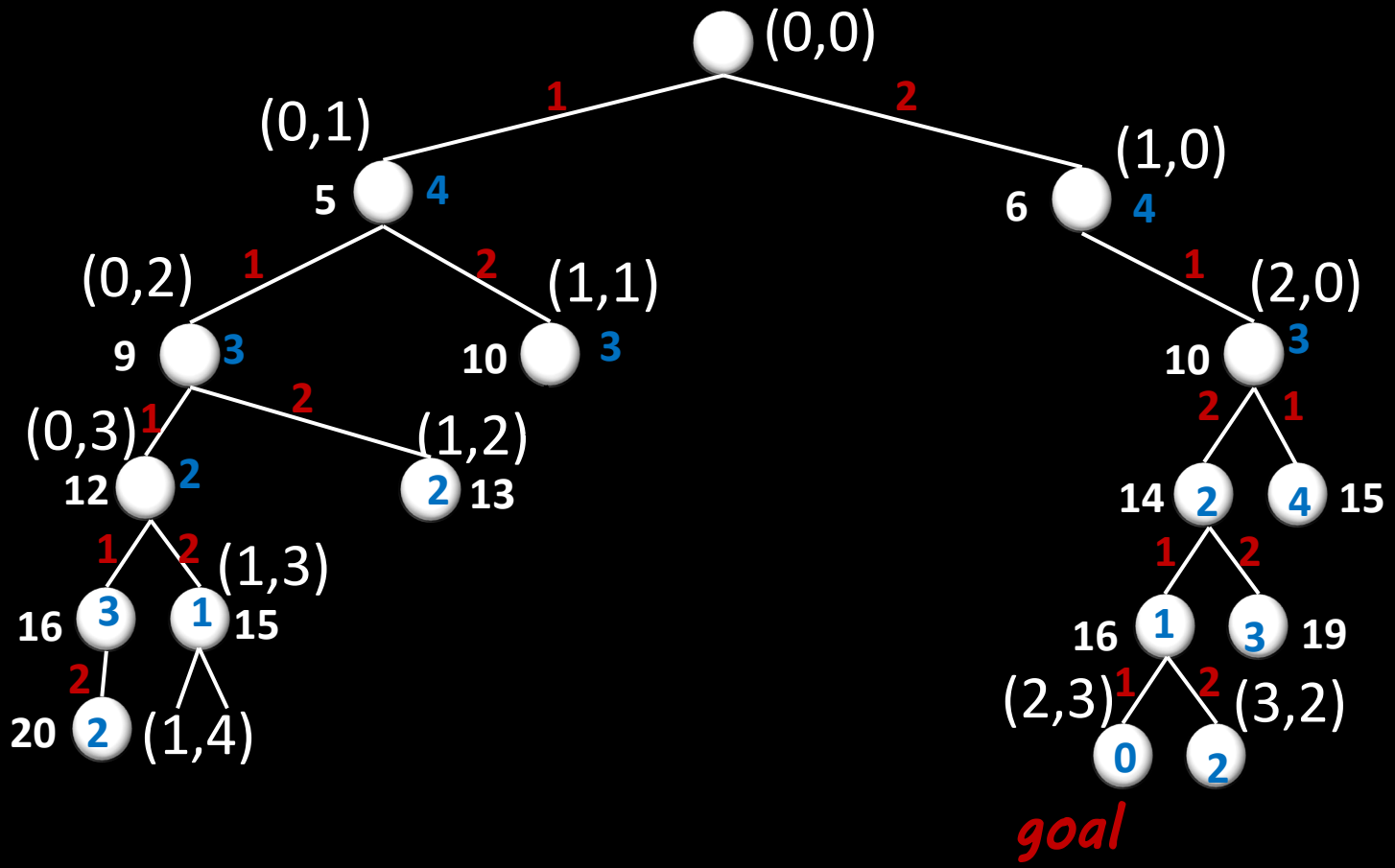
Find a treasure



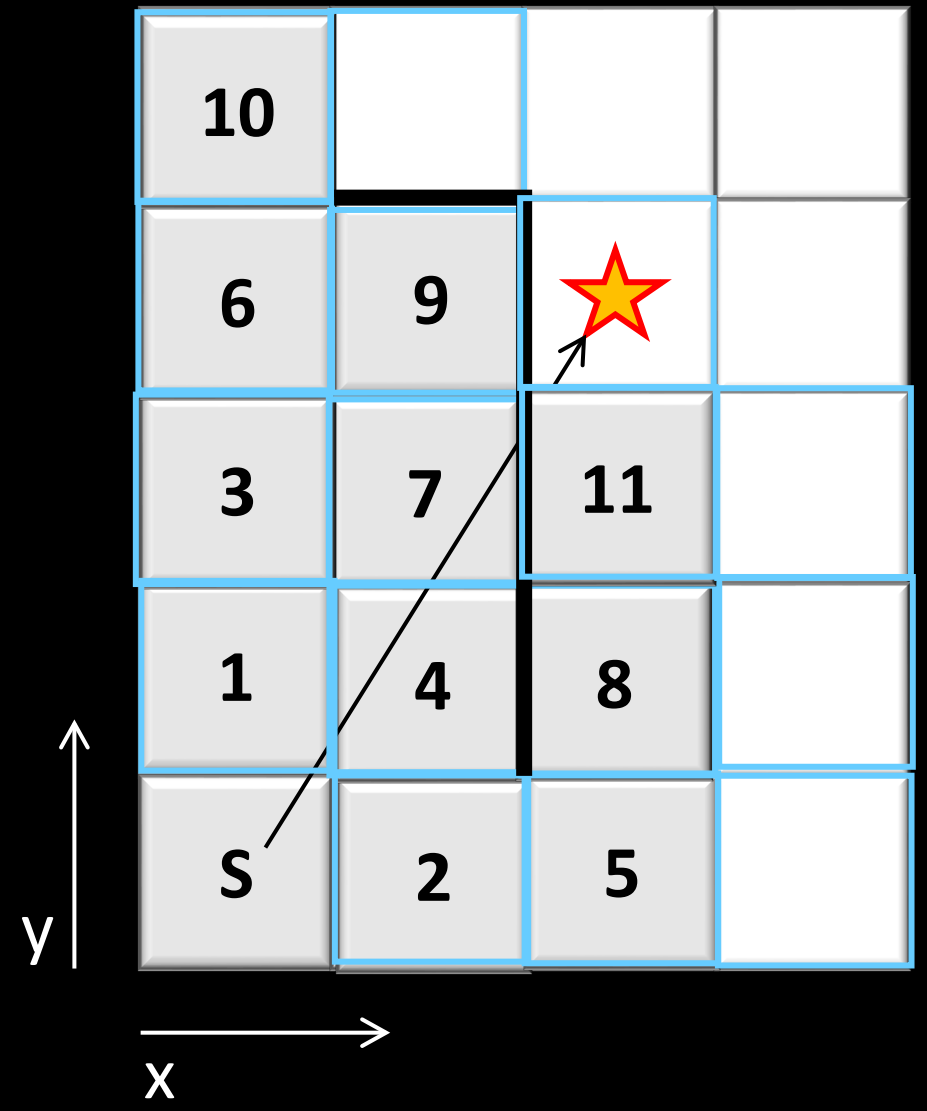
Informed Search

Search order: N, E, S, W

- A* (“A-star”)
  - Cost and goal heuristic



Find a goal



A* Search

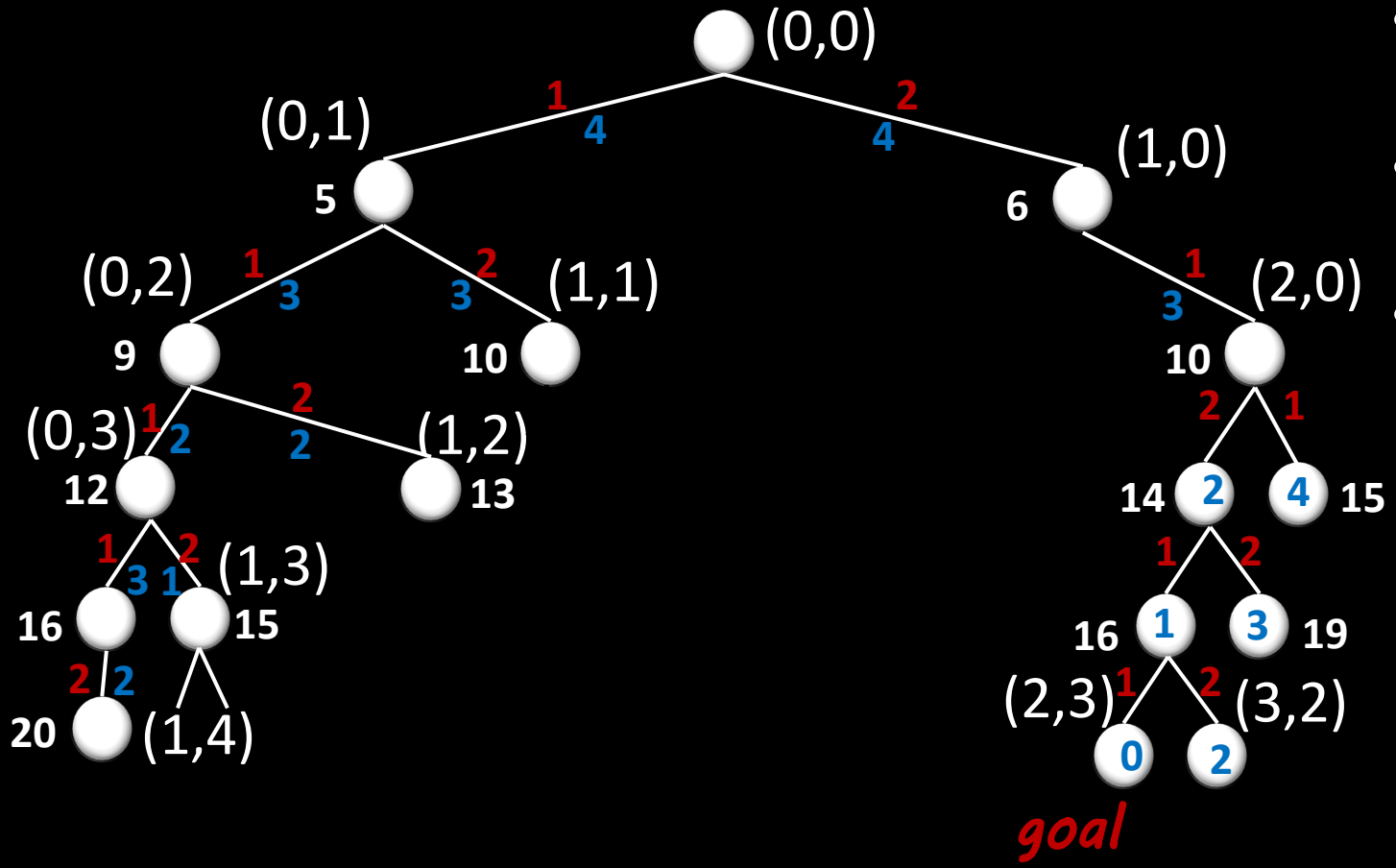
- What if the heuristic is too optimistic?
  - Estimated cost < true cost
- What if the heuristic is too pessimistic?
  - Estimated cost > true cost
  - No longer guaranteed to be optimal
- What if the heuristic is just right?
  - Pre-compute the cost between all nodes
  - Feasible for you?

admissible heuristic

inadmissible heuristic

Informed Search

- A* ("A-star")
  - Cost and goal heuristic

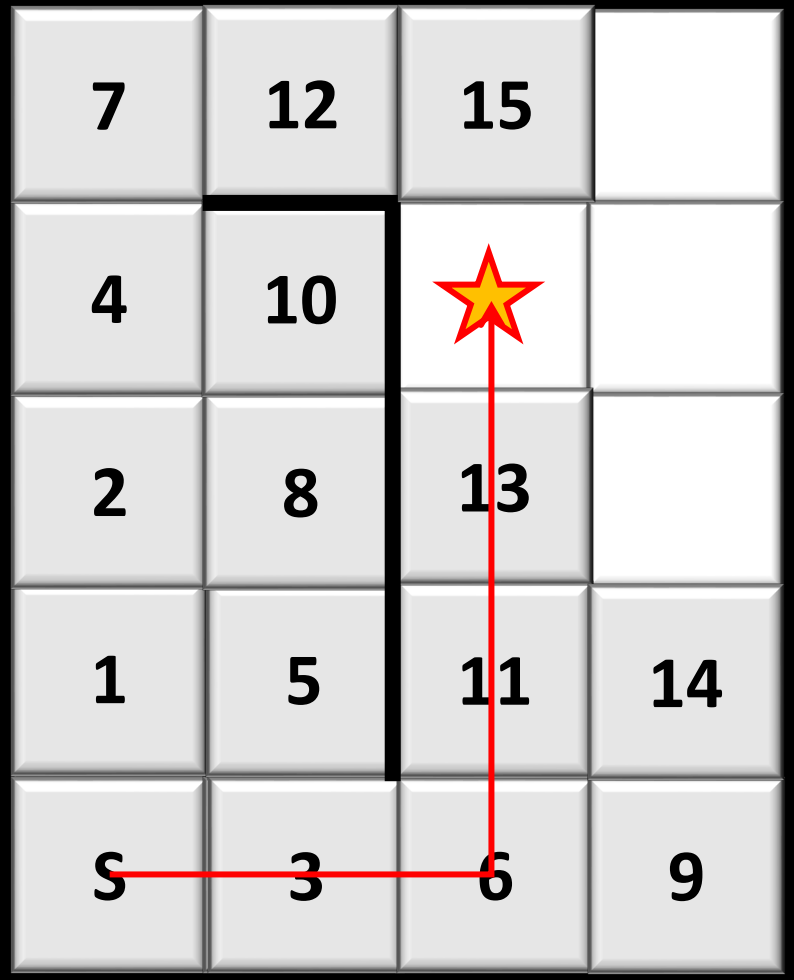


- Complete?
  - Yes!
- Time Complexity
  - $O(b^m)$
- Space Complexity
  - $O(b^m)$
- Optimal?
  - Yes, if the heuristic is admissible!

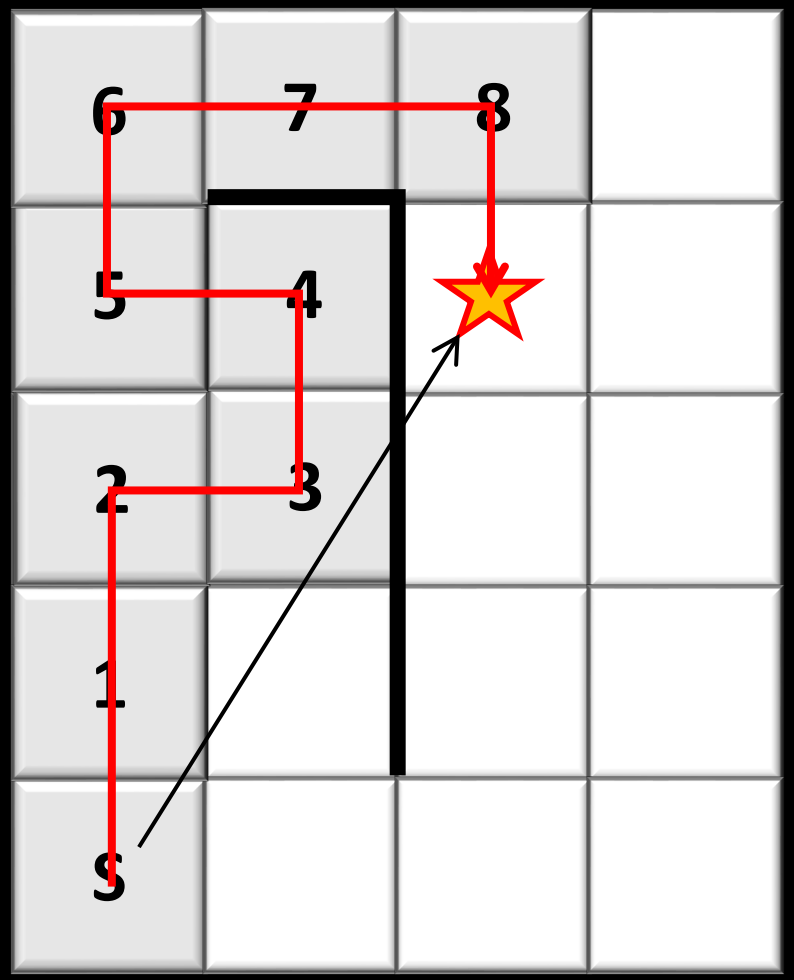
Summary

LCFS

minimum path

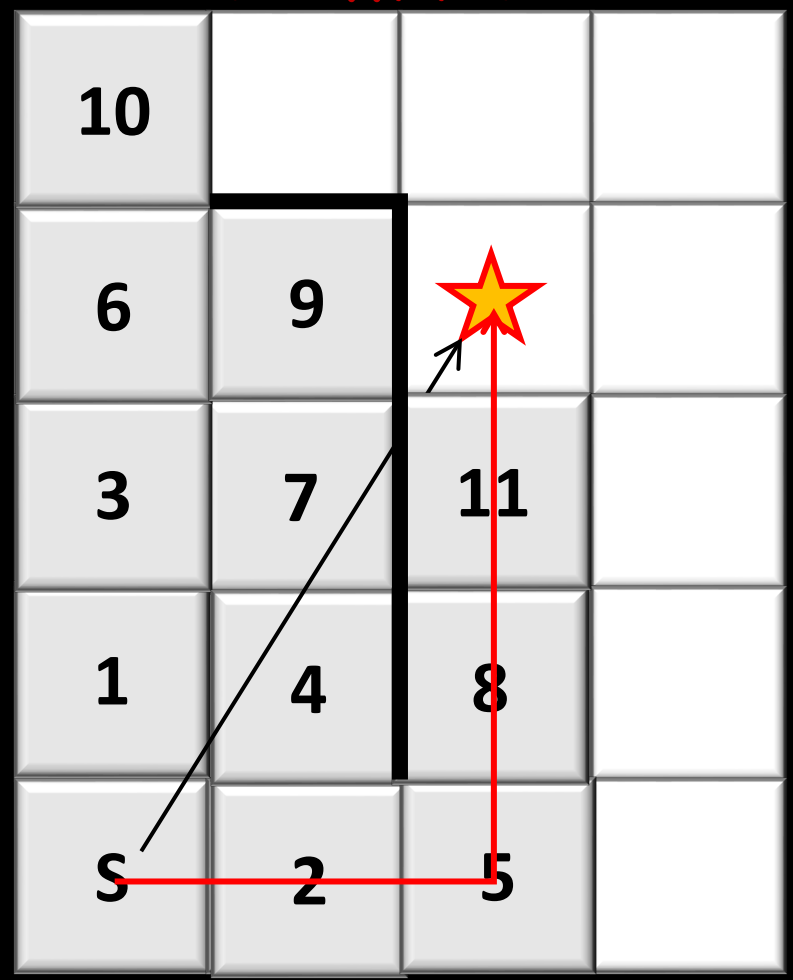


Greedy



A*

minimum path and efficient



Upson Hall, 124 Hoy Rd, Ithaca, N

https://www.google.com/maps/dir/Upson+Hall,+124+Hoy+Rd,+Ithaca,+NY+14850/Cornell+Dairy+Bar,+Tower+Road,+Ithaca,+NY/@42.4452681,-76.4805676,16z/data=!3m1!4b1!4m14!4...

Apps | MPI | Cornell | Passkey | Women of ECE | ECE3400 | NRI NSF Award | ECE 3400-2018 | Piazza ECE 3400 | DARS 2018 | Other bookmarks

Upson Hall, 124 Hoy Rd, Ithaca, NY 148

Cornell Dairy Bar, 411 Tower Rd, Ithaca,

Add destination

OPTIONS

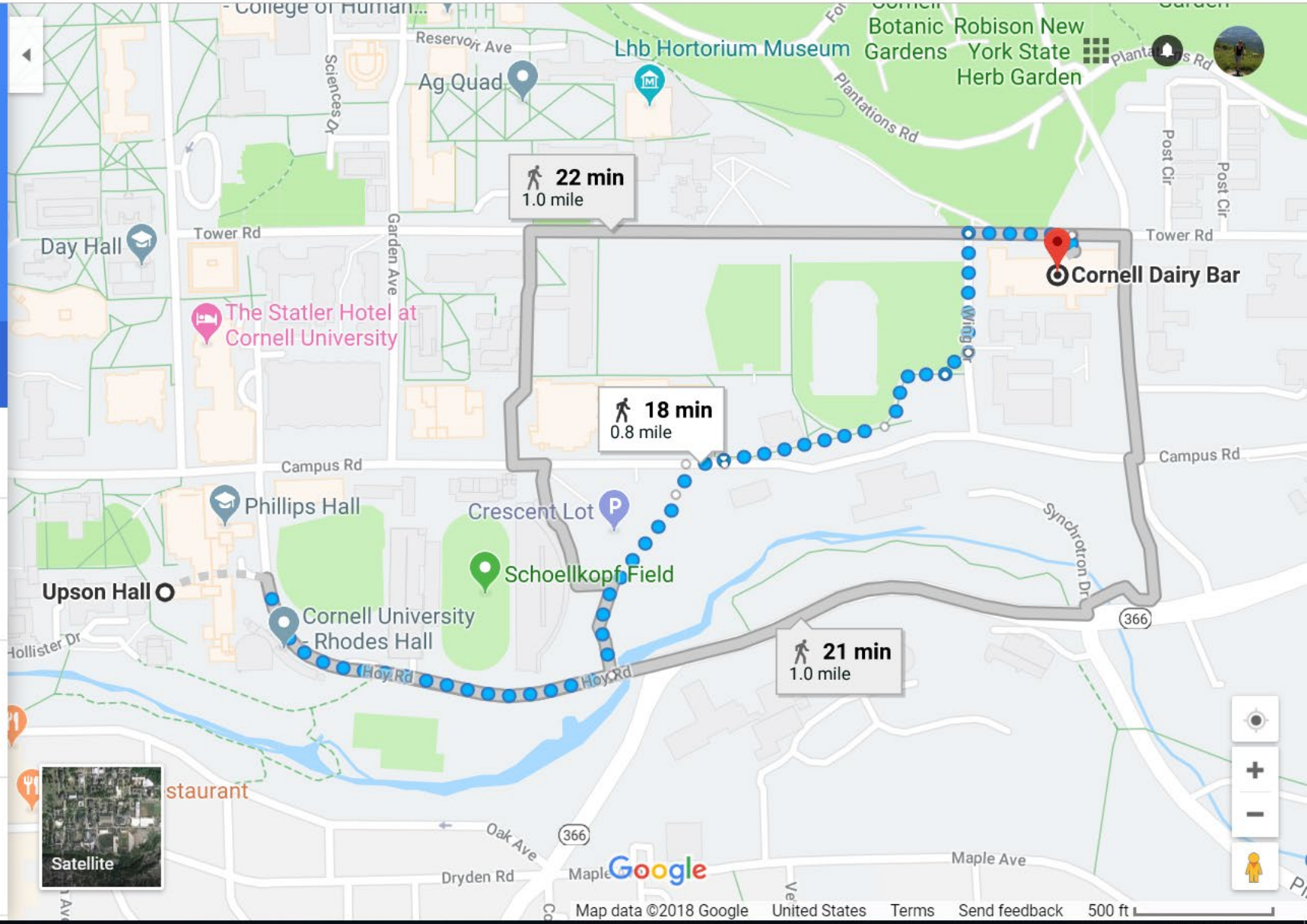
Send directions to your phone

via Hoy Rd — 18 min — 0.8 mile

DETAILS

via Hoy Rd and NY-366 E/Dryden Rd — 21 min — 1.0 mile

via Hoy Rd and Tower Rd — 22 min — 1.0 mile



10:16 AM
10/29/2018